Alessio Venticinque

Semantic Techniques for Verification and Validation of Safety Critical Embedded Systems in Industrial Applications

PhD Thesis in Computer Science and Automation Engineering

March 31, 2014

Supervisor: Prof. Nicola Mazzocca

First and foremost, I must thank my research supervisor Prof. Nicola Mazzocca. Without his support and expertise this thesis may never have been completed. Thank you very much for your encouragement and understanding over these past four years.

I must thank also Ansaldo STS and Intecs companies, for offering me the international experience at Ansaldo STS USA in Pittsburgh and for the exciting participation to the CESAR and CRYSTAL projects.

Last but not the least, I would like to thank my family.

Introduction

The design and development of Safety-Critical Embedded Systems (SCES) is a relevant activity in many application fields such as railway, automotive, aerospace, health, etc. The life cycle of a new product must follow both regulatory constraints and challenging requirements and is continuously affected by a number of conflicting objectives to be achieved such as minimization of costs, improvement of performances, short time to market. Both service levels and deadlines of commitments must be satisfied without failing the required, more and more challenging, safety constraints. The verification and validation represent a relevant phase that both discovers errors and certificates the achievement of the objectives.

This thesis work aims to integrate into the Verification and Validation process of critical embedded systems methods to improve RAMS properties (Reliability, Availaility, Mantainability, Safety) of industrial application. We want to innovate the techniques used during the life-cycle for a railway system, in a world wide company, in order to verify and validate products in a shorter time and with the better results in terms of errors found and evidence of correctness about performed activities.

One of the main issue that affects the development life-cycle of SCES is related to the understanding of specification and analysis of requirements. In fact it is time consuming and depends on the interactions of between heterogeneous stakeholders. Stakeholders include the experts of the application domain and technicians, or anyone is in charge to understand high level system requirements and produce the necessary not ambiguous specification to feed the design and development process. Semantic mismatches, lack of formal

models, the absence of internationally recognized open standards are limiting factors that affect the performance of development process, above all in the test definition phase. In fact they consume about the 25% of the entire testing phase. Many errors occur also in the following activities, for example during the test definition, when it needs to translate the high level specification to formal executable scripts, and they main reason of failures of testing, which sometime imply the revision of the project across the entire life cycle.

Our attention will focus on semantic techniques in order to support requirements analysis and management activities and testing. The presented activities have been development within the framework of two research project that involve industrial and academic partners. CESAR stands for Cost-efficient methods and processes for safety relevant embedded systems. I aims at bringing significant and conclusive innovations in requirements engineering in particular through formalization of multi viewpoint, multi criteria and multi level requirements. CRYSTAL project (CRitical sYSTem engineering AcceLeration) takes up the challenge to establish and push forward an Interoperability Specification (IOS) and a Reference Technology Platform (RTP) as a European standard for safety-critical systems. They are European funded project from ARTEMIS JOINT UNDERTAKING (JU).

The case study belongs to the background production RAMS group of Intecs s.p.a. and Ansaldo STS and in particular to the ERTMS/ETCS (European Railway Traffic Management SystemEuropean Train Control System) railway application. The ERTMS is a system developed by the European Union with the target to remove the technical barriers against the interoperability regarding the train control command system. It define a system of commands, control, telecommunication and management of railway traffic. The ETCS is an ERTMS basic component: it is an Automatic Train Protection system (ATP) to replace the existing national ATP systems. Its basic requirements are described into UNISIG normative; we will give details about both these standards into following chapters of this work.

The approach will be general compared to the single problem. Furthermore, the activities about the requirements management and verification of safety critical systems, applied in railway applications, have to provide the compliance of the software to the requirements, reduce time to deliver and ward off the presence of errors. In particular, we will work on phases of concept of system, verification and validation of software and configuration, log

analysis and support to error analysis according to the current normative. In fact railway system life cycle is defined by CENELEC normative, in particular the EN 50126, EN 50128, EN 50129. These normative define the process and the activities to be taken into account to achieve the RAMS target for the system and in aprticular assure its safety and reliability.

We will apply techniques of requirements analysis in the phases of test definitions, coverage and test analysis. We will introduce formal methods and semantic base analysis. In particular we will use ontology in order to define our knowledge base and to find relation between test scenarios, requirements and also issue discovered. Finally the results will be used to support the user in error discovery and to find relation between same behavior of the system in different test case.

In the first chapter we analyze the issues related the development of embedded system with particular attention to critical applications. Then we will introduce some examples of research effort to use formal methods for the development of SCES.

In the second chapter we present the safety life-cycle currently in use, and the related open issues. We discuss research opportunities in the field of formal methods and semantic techniques to innovate the state of industrial process to support analysis and management of requirements, testing generation and test log analysis.

In the third chapter we propose the application of semantic technique and methods to innovate the industrial process. We present the new work-flow, the techniques to be used and the architecture of a new framework that supports the proposed methodology.

In the fourth chapter we will describe the technologies available for the development of a prototype that supports the user to adopt the proposed innovative approach with their limits and features. The issues regarding the integration with the existing process and CENELEC normative will be addressed in this chapter.

In the last chapter we will present a real case study and the related experimental activities performed as RAMS engineer in the industrial contest of Ansaldo STS and Intecs s.p.a. We will describe the main aspect of the ERTMS/ETCS level 2 demonstrating how the tool works and how it supports the user from the analysis of requirements to the analysis of test logs in the verification phase.

Finally we will conclude analyzing the approach proposed and the results and the discovered limits. The main benefit to the process and future work will be also described.

Napoli, *Alessio*
March 2014 *Venticinque*

Contents

1

Application Context

1.1 Safety Critical Embedded Systems

Embedded system are usually electronic systems based on micro-processors. They are designed and developed to carry out a specific task. In opposite to general purpose machines, the embedded systems will be used to achieve goals which are already known at design time, during the development life cycle. They must be designed to satisfy defined requirements, such as execution of real time applications, respect of memory constraints, high level of reliability and robustness, etc. For this reason the design and development of software for embedded systems must take into account not-functional requirements.

It means that correctness, that means the compliance with functional requirements, is not the only property to address, but it is necessary to check all the defined constraints and the compliance with the related standards. The embedded software is validated more than any other kind of applications because it execute on critical system such as airplanes, cellular phone, medical appliances, energy stations. The quality criteria of embedded software are very strict and it needs a huge amount of testing activities grant that all the critical scenarios are checked without failures, because risk of failures can cause loss of human lives.

The usage of embedded systems is increased a lot during the last years above all in the automotive application domain, due to the spread of electronic control systems in last generation cars.

With the growing of control functions, and thus the more complex the system becomes, it is necessary to adopt robust strategies for managing the software development process in order to reduce any risk factor. In fact it

means a more complex and larger software. Nowadays the lines of code, just for the control of the engine of a car, is estimated to be about 500.000, and can be 1.000.000 if we consider all the functionalities to be implemented. Of course the automotive customers ask for high reliability systems, that is different from what is required by PC applications. The main issue the increasing difficulty to grant, by a brute force testing, a complete coverage of production scenarios with affordable costs, because of the high dimensions of the large number of combinations of states that the system can change.

In the last ten years techniques of risk management have been developed. Some of them include:

- Architecture design and long term high level planning of development.
- Model driven techniques for development of algorithms.
- Model driven automatic code generation.
- Industrial standardization of interfaces and functionalities of software modules for embedded systems .

To address the complexity of embedded system, the best strategy deals with the development of project conceived to be re-used in the future an in different context. A widespread approach to achieve this goal uses a modeling techniques that is component oriented.

A component can be used inside the embedded system only if it does not adversely affect the usual operations and the available resources. When the system has been implemented it need to deal with how it will be tested with a complete coverage. An option would be the formal modeling of the algorithms, before the software design. If the system can be simulated, the algorithm can be tested and validated without coding its implementation.

After that the architecture has been defined and the coding activity is starting, the automatic generation of code can contribute to reduce the occurrence of errors and costs. In the automotive domain, the management of risks, which affect the implementations of control system more and more complex, represents one of main problems addressed by developers.

The different kind of constraints that characterize the embedded systems respect to general purpose software are due to their properties, which change according to the target applications. Hence some issues such as costs, long life cycles, real-time requirements, availability (the probability that a system is

working at a specified time), reliability while a specific function is working, make usefulness the usual techniques for the design of these devices.

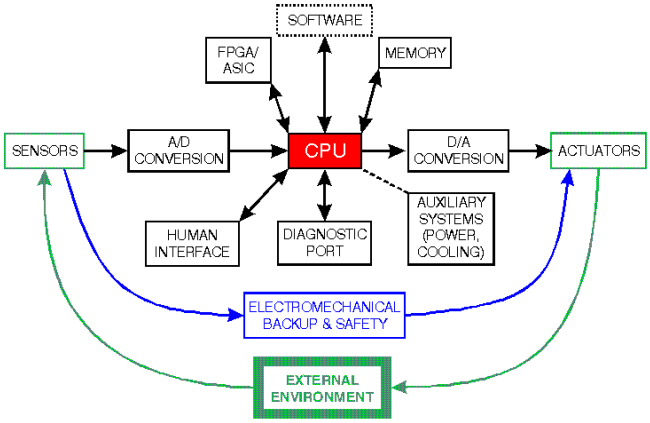

Fig. 1.1. Embedded System

The human interface can be as simple as a light indicator or as complex as the vision of a real-time robotic system. The diagnostic interface could be used to monitor the system to be controlled like for an engine, but could not suited for the diagnostic purpose of the same embedded system, FPGA and ASCI technologies are used to improve the performance of the system. The software implements management and control functions. Sensors, actuators and converters allow to the embedded system to communicate with the environment.

Embedded system must guarantee both functional and technological constraints. In particular the must be reactive to external stimuli and the compliance with space constraints (both hardware and software). Moreover the must satisfy safety requirements and reliability, without exceeding the planned budget.

1.2 Critical Infrastructures

The first definition about critical infrastructure it has been reported in the Marsh Report (1997): "a network of independent, mostly privately-owned, man-made system that function collaboratively and synergistically to produce

and distribute a continuous flow of essential goods and services ". The Europe Community provides a definition about what are critical infrastructure in the directive 2008/114/CE: "critical infrastructures means an asset, system or part there of located in Member States, which is essential for the maintenance of vital societal functions, health, safety, security, economic or social well-being of people, and the disruption or destruction of which would have a significant impact in a Member State as a result of the failure to maintain those functions.

Regarding the information and communication context (CI) some infrastructure highly sensitive or "critical " represent the state itself and ensure its operation (the institutional settings), others are essential to everyday life , such as electricity and water , and others again are part of the fabric economic and social development of a Country, such as banking and telecommunications. Public transportation is perhaps the biggest challenge for those who have the responsibility to monitor safety systems. The extensive infrastructure (think about a large metropolitan city center), the highly populated area, highly sophisticated and expensive vehicles in continuous motion require a global solution that guarantees a management strategy that supports the highest alert level.

In railway stations it is mandatory to monitor 24 hours a day the access control, the movements of travelers and lost luggages, and at the same time it needs to monitor physical structures arranged in wide geographical areas (tunnels, rail crossings, lines, platforms , electric wires), warehousing (deposits of materials in transit , power stations and rail centers).

Critical infrastructure can be divided into different classes of interest. The classification is not unique and depends mainly on the impact they have on national requirements:

- Production plants, transportations and energy distribution (power grid, pipelines, etc.).
- Infrastructures for communication.
- Banking and financial systems.
- Health-care systems.
- Infrastructures for transportation (air, road, rail, port, etc..).
- Infrastructures for the collection, treatment and distribution of water.
- Emergency services.
- The food supply chain.

1.2.1 Safety of railway systems

The operation of a railway system is very complex because it needs to manage all the entities installed in line and to guarantee the correctness of their functionality, together with service levels and safety requirements. All the train present in line have to ride in a condition of safety, keeping their speed and the distance to others train compliant with the braking performance. Considering the high speed of the trains and the high capacity of the line, that the market requires, it is impossible to entrust all to the human work. However also the project of such systems is not free from errors, especially when new features must be provided within strict deadlines.

We recollect on Saturday 23^{th} July 2011 [32] a tragic railway event striking the Chines high speed railway line. One train, stroked by a lightning, remained stopped on a bridge in the location of Zhejiang for damages to the electric equipment. A second train came in the same direction and with high speed and it crashed into the stopped one causing the derailment of four carriage and the fall down of two others. In the incident 43 people lost their life and wounded were over 200 on a total of 1400 passenger, 600 of them were in the wagon involved. The following investigation activities have shown that the cause of the accident was related to an error in the design of the signaling equipment. This error avoid the second train to receive the stopping message instead of the free one (probably something happened like a green lights was showed instead of the red one).

This example shows us how a simple error can have catastrophic results. Even if the used techniques have guaranteed good results for long time, their application to the design of modern systems affect the cost, which increase exponentially. As usual the main issue deals with the compromise between costs and service level. Passengers are interested to service level which shall be always better and at lower prices, but safety cannot be less than in the past. On the vendors side, service providers must be offer competitive price to avoid of being cut off from the market. However this problem could not be resolved at any cost. The amount of investment needed to improve the safety have to lead to saving money compared to the compensation for deaths and injuries; otherwise the safety performance is not sustainable. It is all how much the company want to risk.

The risks are measured by the number of weighted fatalities (FWSI) per train kilometers. The FWSI are the fatalities and weighted serious injuries (amputees, burn victims, crushed) and is calculated in equation 1.1.

$$FSWI = 1F + SI/10 \tag{1.1}$$

Where F stands for *fatality* and SI for *Serious injuries*.

Even if this formula seems to reduce the life of people to a merely economic issue, the results achieved in the years is very good if compared to others means of locomotion. In Table 1.1, from [11], the fatality risk of passengers using different mode of transport is showed. According this report airplanes and trains are the most safety transportation, whereas those preferred by Italian people are the worst.

Transport Mode used by User	Fatality risk (Fatalities per billion passenger kilometers)
Airline passengers	0.101
Railway passenger	0.156
Car occupant	4.450
Bus/Coach occupant	0.433
Powered two-wheelers	52.523

Table 1.1. Fatality risk of passenger using different mode of transport (EU-27 in 2008-2011)

In particular it must be noted that in the EU-27 the number of railways passengers killed in accidents in 2011 are 38 against the 30268 of the road fatalities.

The high standards of quality and safety of the railway systems need the support of methodologies and instruments that allow the company to furnish a modern, or better, innovative product with a competitive price. To achieve this target the company from one side has to evolve the products innovating in functionality and technologies, but on the other side there is the necessity to update the process in order to manage the growing complexity of the system, reducing the effort needed for some activities and moving the gap to others more relevant.

1.3 Safety Life Cycle

The Safety-Life-Cycle (SLC) is a closed loop process. Its activities are developed continuously till when the system is in production. The activities are extended and more intensive when the system is changed or updated. The kind of process is used in many Italian and European directives (e.g. IEC 61508, EN50126). The common Safety-Life-Cycle can be organized by three main phases:

Analysis : identifies the dangerous events and risks, their probability of occurrence, consequences and all the activities that can be done to avoid or mitigate them.

Realization : includes the development and construction of the system.

Operation : the system during its commissioning, maintenance activities, system update, and finally its disposal or replacement.

In Figure 1.2 the closed loop SLC is shown. The three different circles represent the three main areas described before. The life cycle defined by the IEC 61508 directive covers all the SLC areas. For each of the it defines the activities and the steps to be done, the necessary information to develop each step and the documentation to be produced. The analysis include the Concept, the overall scope definition, Hazard & Risk Analysis, the Overall Safety Requirements and the Safety Requirements Allocation. The Realization Phase is composed of the Overall Planning, Overall Operation & Mantainence Planning, Overall Safety Validation Planning, Overall Installation & Commmissioning Planning, Realization Safaty Related System EEPES, Realization Safety Related System Other Technology and Realization External Risk Reduction Facilities. The Operation Phase is divided into Overall Installation & Commissioning, Overall Safety Validation, Overall Operation, Maintenance, Repair, Overall Modification & Retrofit and Decommissioning or Disposal.

1.3.1 Analysis Phase

The analysis is a critical activity of SLC. It can be developed according to the following steps:

Determination of Acceptable Risk : the client has to decide what is the risk acceptable for his system . This risk should be compared with the intrinsic

Fig. 1.2. Safety Life Cycle

one of the application to assess whether it is appropriate or it should be changed.

Hazard Analysis and Risk Analysis : it consists of a classification of all the dangers and analyze the risk associated with each of them. This activity includes the identification of all the functions necessary to detect dangers and to put the system into a safe state .

Documentation : it represents the collection of information that are the output to the analysis phase . This information is contained in the specification of safety requirements (SRS). The goal of this specification is defined in the standard as *the aim of developing the specification of the global safety requirements, in terms of the requirements of the safety functions and safety integrity requirements , systems security-related E/E/PES for safety-related systems based on other technologies, and for external devices to reduce the risk, so as to achieve the required functional safety*

1.3.2 Realization Phase

The realization phase begins with the design of the system in order to meet the specific safety requirements, and then continues with the build, installation and validation of safety-related subsystem as specified in the previous phase : First it needs to choose the technology and the system architecture required to meet the specified requirements. The choice is made according to the Hazard Analysis contained in the SRS. Once the design of the system has been completed it should be checked that it meets the levels of SIL described into the SRS. The detailed design should be performed in accordance with clearly and well defined established procedures. Finally the installation of the system, the deployment and the validation stages are completed. The system thus installed will work at the level of risk that has been accepted and established.

1.3.3 Operation Phase

The operation of the system start after the validation of the project. This phase consists of the maintenance and diagnostic tests about the safety of the system. In fact, the SIL of the system can be influenced by the number of times in which the system is tested and repaired to keep the condition of full functionality. The organization responsible for the operation of the system must correctly analyze the effect of the changes and the of deactivation of the safety mechanisms on the process and on the plant under control.

1.3.4 Application of the model

The application of the model must be adapted to the target system. In particular for each case study it needs to specify the phases of the life cycle for the target system and demonstrate that they will be compliant with the directives. The system supplier must define a RAMS program that facilitate the achievement of safety requirements. A RAMS program is usually built on a model that comes from other RAMS programs which are already in the operation phase and from the requirements of supplier's system. The procedure for definition and execution of a RAMS program can be described by the following activities:

- Definition of a life cycle that is feasible both for the system and for the supplier.
- Association of each stage of the life cycle to a RAMS phase and definition of the tasks to adequately cover all the requirements of the system.
- Identification of internal and resources to which the responsibility to perform the RAMS tasks will be entrusted.
- For each RAMS task procedures, tools and documentation to be used must be defined.
- Application of the RAMS process within the company.

1.4 Verification and Validation

The target of the verification is to give evidence that for the specific input of the activity, all the output elements fulfill the requirements of that activity. Instead the aim of the validation is to demonstrate that for each time of the life-cycle the system meets all the requirements specified for it.

The development process of safety critical software must involve peoples who have been well formed and motivated. They must be skilled on the application domain, on software engineering, computer science, safety, normative and regulations. The software validation must be performed by an entity that is independent from the one that carried out the development. Their independence does not mean that they belong to different companies, but it is defined by some parameters. Such parameters depend on the kind of SIL (Severity Integrity Level) it needs to achieve: In the case of *SIL 0* there are not restrictions. Regarding to *SIL 1* and *SIL 2*] Verification and Validation can be carried out by the same person, who must be different from the designer/developer. More complex is the case of *SIL 3* and *SIL 4*. In the case the software should guarantee an integrity level 3 or 4 two different approach can be adopted:

- The V&V activity can be delegated to the same person, who must not be neither the designer nor the developer and must be independent from the project manager. A very important feature is that for a software with this integrity level the organization entrusted to conduct the V&V activity must have the right to deny the software release.

- The V&V activity is entrusted to different persons/groups. The designerdeveloper and the verifier must be different, but they can both depend on the Project Manager. The validator must be either a different parson and independent from the Project Manager and must be the right to deny the software release.

The validator can be an employer both of the organization that provide the software and of the client organization. Even if it belongs to the client or to the supplier company the following properties of the validator's profile must be granted: authorization from the Safety Authority, complete independence from the project team, direct reporting to the Safety Authority.

1.5 Business requirements

The always more competitive of the global market and the speed of the technology evolution bring the companies to assure a products of high quality and with low cost. This is very important especially in the development of the safety critical embedded system where a low quality could bring to an unsustainable service and to critical accidents. In such scenarios the companies' target is to improve the quality and the technology content of the products, but at the same time to reduce the costs of the activities in terms of effort and time.

The engineers try to reach the result following two different way. One is to modify the process in order to anticipate problems and do not wast time doing not useful activities. This issue affect in particular the verification and validation activities. In this phases the system developed is under inspection in order to find errors, to prove the compliance of the system to requirements and to show that the quality standard are achieved. Often the activities performed in this phase are composed by a set of work item that have to respect a fixed sequence and cannot be executed in parallel.

The field test are executed in the real environment, with all the target component including the train that ride on the binaries. Suppose that all the test are to be executed in field and the test set in order to verify and validate a baseline is composed by 1000 test case. The test throughput is about 15 Test/day and the cost of each day of test is 7.000 . So the time to complete the activities is 66,7 days with a cost of 466.666,67 . Usually the baseline to

test before to start the service are about 6; totally we spent 2.800.000 , which are a lot of money. The factory test are executed in laboratory using simulation environment on a general purpose machine, or a target machine, but simulating the environment that is interfaced with it. Introducing the factory test we have to perform the test twice, but only few baseline will be tested in field. Considering 100.000 the amount to develop the test environment and 1000 the cost to execute 30 factory test in a day. So the time and the price to complete the activities for the 6, as mentioned before, baseline is 200.000 in 200 days. The factory test allow also to reduce the baseline to test in field to 3, the resulting amount in this case is 1.400.000 (1/3 of the precedent). The total amount, after introducing the factory tests is 1.500.000, saving 1.300.000 from the case 1.

So what in a first moment seems to be an extra amount, allow to improve the time to market and the price.

1.6 Formal Methods for Safety Critical System Life Cycle

Software engineering use formal methods to the specification, the system design, development and verification. The Encyclopedia of Software Engineering [24] defines the formal methods as follows: *Formal methods used in developing computer systems are mathematically based techniques for describing system properties. Such formal methods provide frameworks within which people can specify, develop, and verify systems in a systematic, rather than ad hoc manner.* It means the formal methods provide a frameworks that support users to specify, develop and verify systems adopting a systematic approach. In the specification of requirements phase[35]formal methods are used to describe in a not ambiguous way what the software will do and how. They aim at avoiding the developer misunderstands the specification and writes code that is not compliant. During the verification phase formal methods are used to check and prove that the system that has been built is the right one, that means it satisfy all the requirements which are formally defined. For systems with a limited complexity, where it does not need to satisfy strict not-functional requirements, and it is not necessary to guarantee fault tolerance, semi-formal methods can be used. For example for the analysis and design of these systems diagrams (UML), tables and simple text notations are used, but there

are few mathematical features as baseline. This is not suited for systems that are characterized by high complexity, which must guaranty very strict not-functional requirements. These kind of systems must address dependability such as security, reliability, performance, real-time responsiveness, availability. Other constraints can refer to a limited amount of available resources, such as limited memory, limited number of concurrent tasks, low energy consumption, etc. For these kind of system is hard to work without formal methods. In fact, in order to support the verification of such properties it needs to define the functions and the behavior of a system using formalisms that allows for both a semantic and a syntactic description. On the other hand the adoption of such techniques introduces an additional cost due to their complex usage and to a longer project schedule. This overhead can be leveraged by the usage of tools which hide to developers the complexity of mathematical models which are the baseline of the formalisms.

The first issue addressed by who is defining the specification, or is describing the system architecture is the matching between the real world and the formal model. A second issue is the complexity of the solver for the target model. Moving from the design phase to the development phase the system model must be translated into a software implementation, which must be compliant to the original model by maintaining the same properties. Another issue to be considered is related to the errors, which can affect both the model and the software. The system abstraction can be wrong or incomplete. The wrong model can be due not to an error in the modeling phase, but it could come from a wrong analysis or definition of requirements. Using testing it is possible just to find and demonstrate the error occurrence, nothing can be granted about their absence.

1.6.1 Formal Specification

To better understand how much formal methods to introduce improvements in the software development cycle, we must first consider the shortcomings associated with those who lack a formal approach:

Contradictions : specifications which contrast with each other, for example in a condition it is said that in case of locking of a wheel the ABS must release the brake, in another that the wheels must never be blocked.

Ambiguities : definitions that can be interpreted in different ways.

Vagueness : vague definitions and lack of information. E.g. : *The display must be user frendly.*

Incompleteness : This is one of the most common mistakes, and when it occurs when a requirement has been described without precise information: *The light should blink for a certain number of seconds.*

Using formal methods for the preparation of specification of requirements allows for the definition of the system in a precise manner, to eliminate ambiguity in its definition. In this way in the next coding phase the design choices are implemented correctly; vice versa building software on a wrong specification will be surely wrong. Writing the specification, in this way, helps to clarify the requirements, to find latent errors, ambiguities, and to make decisions about the functionalities at the appropriate level of the development process. During the specification phase by formal methods it is almost impossible not to make comprehensive decisions on functionality, because this results in a specific non-coherent, whose errors are immediately detected by the developer. The usage of a formal syntax for a specification language allows the requirements or to the design specification to be interpreted in a unique way. This eliminates the ambiguity that often occurs when a natural language or diagram are interpreted by a reader. The completeness is still a property difficult to achieve even when using formal methods. Some aspects of the system may be not defined when the specification phase starts. Other features may be omitted on purpose to allow let the designer free when approaching the implementation, and it is impossible to consider every possible scenario for all operations in a large and complex system. Finally things can simply be omitted by mistake.

1.6.2 Formal Verification

The verification of the correctness of the implementation is a particularly delicate and expensive phase. Reliability and quality of the product depend on this activity . In principle, the verification was done exclusively through black-box and white-box testing techniques. In particular, the first kind was used for the verification functional requirements and for checking the correctness of interfaces. The second one is used for structural errors. Later simulation techniques have been established , and more recently ormal verification techniques, based on the use of special logics, have been proposed.

Formal verification is often preferred to simulation because for complex systems the last one is not exhaustive and it is very difficult to define test patterns that prove the correct operation of the software.

By formal verification techniques, instead of having a trial or an experimental proof, we have a mathematical proof. The formal verification techniques can be divided according to the degree of abstraction and the type of mathematics used. The verification can be performed at two different time of the development process:

A priori : during the design phase, granting the correctness of each development step;

A posteriori : at the end of the implementation.

In order to apply these techniques the system that we consider should be expressed in a mathematical logic. Normally it is one among First-order logic, Second order logic and Temporal logics.

The first two languages refer to theorem proving techniques. Temporal logics are instead used by model checkers. A Posteriori verification is based on a formal theory and a theorem prover or model checker based on it. The verification may be divided into three different steps:

- Translation of the specification into a chosen language.
- Drawing up of a set of axioms or lemmas to be used for verification.
- Identification of verification strategies, according to the abstraction model.

The model obtained from the system have to be more abstract, otherwise you run the risk that state space to be explored is too large, there also needs to be focused only on certain aspects of the software. Once you have the model you are using one of two techniques mentioned above.

1.6.3 MDA

The Model Driven Architecture MDA is an approach that supports analysis, application design and developments(implementation), but also testing, deployment and maintenance. The MDA is an Object Management Group (OMG) standard implemented as an architectural framework for software development in order to reduce complexity, costs, and improve the development of new software. The MDA supports the use of system models in the life cycle

and the reuse when set of systems of the same domain are created, the enterprise architectures are supported by automated tools and services to design the models and to allow the transformations between different type of them.

This standard should not be considered a real specification, but it is an approach to the software development that relies on the others OMG specification like Unified Modeling Language (UML), Object Facility (MOF) and Common Warehouse Meta-model (CWM).

The MDA aim is to support the development of flexible models that could be machine usable and can address the following topic:

- Technology obsolescence: the presence of a design supports the integration of new infrastructure in the existing one
- Portability and Integration: the existing features can be migrated to others platform in easy way and are facilitated the development of links with others systems or components.
- Productivity and time-to-market: the efficiency is improved by automating the activities tedious and repetitive.
- Quality: is improved by the consistency and the reliability of the artifacts and the independence of concepts.
- Maintenance: the design representation in an machine usable format helps the stakeholders to address the issues linked to the maintenance.
- Testing and simulation: the coherence between requirements and models can be validated in easy way, furthermore the model can be used to simulate the behavior of the system.
- Return on investment: applying such standard and investing in tools can bring a huge profit.

In the MDA the models are a key factors to understand the systems for enterprise-scale solutions. The systems developments is based on a set of models and a set of transformations between them, splitting them into an architectural framework of layers and transformations. In order to support the transformations and the integration thorough automatic tools the models are described by set of meta-models. A key solution for the wide spread of this model based approach is the definition of standards for tools integrations and competitiveness. These targets are achieved through a set of layers and transformation defined in the MDA standards together with the identification of four type of models:

- Computational Independent Model (CIM): this is a business model which vocabulary is that of the domain expert. It is used to specify what conceptually the system do, but the information related to the technology and the implementation are not present.
- Platform Independent Model (PIM): it is an high level model that can be applied to different types of platform. The system is described trough a set of services, but without describes the technical details.
- Platform Specific Model (PSM): describes how the PIM is particularized in a particular platform. However the PSM does not identify all the details in order to produce an implementation of that platform.
- Implementation Specific Model (ISM): it describes the specification with all information to develop a system for the specific platform.

The models and the software can be classified in relation to the details that they give about the target platform. Such details are for example the hardware, the language, the communication protocols and so on, and represent the elements for a solution platform.

Another key issue is the definition of what models should be used in the different levels of abstraction, remaining independent from the definition of the platform. Finally there is the definition of transformation and refinement between models. To perform this activities is required the knowledge about the domain ad the characteristic of the specific implementation as technologies. In order to improve the efficiency the transformation should be reused for more models. If we consider the activities of system development as a sequence of transformation and refinements, it is clear that this becomes the main issue of the development process.

2

Testing and Validation of Safety Critical System

The testing and validation of safety critical embedded systems requires a lot of resource in terms of time and people involved. At the state of art, available methodologies aim at improving both the coverage, the time spent and the cost of the complete testing and validation process.

2.1 A general model of testing process

To define a general model of testing process it is possible to identify the following activities: *test definition*, *test execution*, *test report analysis* and *test report document drawing up* (a document filled on the basis of test report analysis). In particular, the technical gaps identified in each activity and the amount of effort spent to develop each phase exploiting current techniques and methods are illustrated in the Table 2.1. The amount of effort put on each testing activity is indicated in percentage points where 100% represents the current overall effort spent in the testing process.

2.1.1 Test Definition

The test definition consists of the specification and development of test cases, which have to be executed in simulated environments; manually by an user or in automatic way. In this case the test procedures are translated into a test script using a proprietary data format. The main issue of the test definition is related to the description of requirements, which is always in natural language and only in few cases it is furnished together with diagrams or UML Models.

Table 2.1. Effort spent in the in each activities of the test process

Activity	Technical Gaps	current effort
Test Definition	Starting from system requirements, tests are manually defined and recorded in test cards. Then, the test cards are used for a manual execution or translated in test scripts, in a proprietary data format, for an automatic execution.	25%
Test Execution	In the "current practice", where an automatic test execution environment is available, there are interoperability problems due to different proprietary data formats from heterogeneous providers. In particular, test data and test logs are expressed in a proprietary format, usually different for each provider.	15%
Test Report Analysis	Most of the efforts spent in this phase concerns the test report analysis that is manually performed.	50%
Test Report document Drawing Up	Most of the efforts spent in this phase are due to the test report document that is manually drawn up.	10%

The specification and development of test cases from such specification is very difficult and it is affected by errors derived by incoherence, incompleteness and lack of clarity. Also the test cases are written in natural language, but they are usually completed with a detailed picture to specify and clarify the description. However the obtained procedure is not free from the same errors that affect the requirements and does not support the use of automatic methods to produce test scripts.

Nevertheless these problems and limitations, a great effort has been spent in order to improve the development of test scripts and their reuse, saving time and costs on these phases of the verification and validation process. Since the tools and the methodology to speed up the test scripting is not of great interest and it is related to proprietary format used inside the company, we will focus on the methodology to support the reuse of the test from a project to another.

2.1.2 Test Execution

The output of the test execution is a set of test logs that include the report of test script execution, the monitor of all variables about the simulated subsystems and the output of the stimulated one. The log is the output of the executed script, in which time-stamps and the value assumed by variables to be checked are reported. Therefore the log contains the state, the input sequence as reported into the original file, but with the time stamps. For what concerning the output sequence and the associated checks, the time stamp and the value assumed by the output are reported and, in case of error, the notification of failure. The checks operate also as break point, if the result of the test is False, the next sequence (input or output) is not scheduled. The report of the executed script is a simple way to verify the test result and the termination mode, but an accurate analysis is essential, by an investigation about what happen in each subsystems, in order to understand the cause of failure.

The tests are executed into a simulation environment in a distributed system. On different servers the machine hosting the simulation environment are replicated. Each machine executes only one test at time and, when finished, returns the result to a service manager. The user submit the test or the test set to the dispatcher and it is responsible for the management of the workload on the different machines and for the collection of the test logs and of their results.

2.1.3 Test Report Analysis

While the reuse of the already available tests is improved using the configuration of abstract script, the analysis of the test log result is expensive and committed entirely to the user without supporting tools. All failed logs have to be analyzed and re-executed in order to prove the correctness of the modification to the software and the lack of regression. Actually, the average effort, using the current test equipments for a typical industrial project (more than 2000 test cases generated from almost 1000 requirements), concerning test definition and test script translation phase, is shown in Table 2.2.

Due to the pace of change is quickening, funding is reducing and the emerging economy are growing their competitiveness, it is essentials to achieve more

Table 2.2. Testing activities

Activity	Average rate	Time consumption (1 person)
Test Cards definition from requirements in Natural Language	40 test/month	50 months
Test Cards translation in Test Script	300 test/month	6,7 months

with less, and at the same time deliver ever better quality. The unique solution that a company can adopt is get better its efficiency and supply always better products, then the above mentioned activities have to be improved.

2.1.4 Test Report document Drawing Up

The aim of this phase is to produce all the documents needed to give the evidence that the verification activities about test results are performed. In such documentation, e.g. the test report, the test executed set is described together with the results of each test for each baseline. To perform this phase the test log analysis is imported in order to explain, for each not passed test, if there are some non conformities and which impact they have. For example, for all failed test linked to an implementation error, it is declared if the anomaly is regards the safety, the reliability or the availability properties.

This activity is completely entrusted to the user, who is in charge to collect all the information, from the test plan to the test results, and to draw the documentation. Since the analysis has just been performed in the previous phase the effort spent in this activity is not much. However the document drawing up is affected from other issues regarding the adherence of the documentation to the company's standards format and the time spent to collect and classify the input artifacts. Hence this activity could be improved using automatic report generation methods. They force the user to introduce all the information in the right way and to export them in the desired format, according to a standard template, or email them to a distribution list.

2.2 Preliminary Considerations

We aim at looking for solutions to the critical issues that beat testing and requirements analysis of Safety Critical Embedded systems during their development. In particular we have to start from the test definition activity that is affected by the usage of natural language for the requirement specification in terms of effort, correctness and coverage. After we will move towards automatic test generation and automatic report composition to investigate the traceability of a safety critical system. In particular we will focus on advanced techniques for knowledge management across the phases of the general model of testing process presented before. We will investigate the integrated utilization of semantic techniques and text search engines for supporting the users to improve traceability of requirements and test descriptions. It aims at leveraging the task of mapping between tests and requirements to represent the coverage of the system functionality. It also allows for an advanced discovery and retrieval of test descriptions and requirements.

2.3 Research efforts for innovation

Many scientific contributions provided in the recent years by the research community is related to improvement of the testing and validation process by the development of intelligent techniques for the management of the knowledge base, composed of requirements and test cards, which are usually written in natural languages. Semantic techniques and formalized ontologies are used for building structured information and for inferring implicit knowledge from the document base, which can be exploited by automatic tools. It allows to speedup the different phases of the testing process and to complement the expertise of engineers.

2.3.1 Knowledge representation

In the process of development reliable and safety embedded systems a key value is the experience that the engineers have about the particular domain. If we want to automatize the knowledge management, to overcome the limitations presented in the previous paragraphs, the experience have to be captured from the people and represented in a form of knowledge that could be used in automatic processing like those presented above.

Here we focus on related work about the knowledge representation in the railway context as it is a relevant domain for the application of testing and validation methodologies, from which we will borrow the case study of our experimental activities. The success to increase railway market in the next future depends in the capability to satisfy the expectation of service quality by the end-user in terms of availability, time to destination and reliability of the service. Since the railway systems are become more and more complex in order to control the complexity and allow further performance improvements, it is necessary to support interoperability between different actors in railway domain through standardization and information sharing. The INteGRail project[17] defines a methodology and a technical platform to support the interoperability inside and between the main areas of railways systems. This target is achieved through a standard language, a standard protocol of communication and a standard meaning of the concept used by actors in order to avoid ambiguity and misunderstanding. The InteGRail is the most important railway project that use an ontology based system approach, its architecture is based on Ontology and SOA (Service Oriented Architecture). The railways domain considered by the INteGRail project was divided into four main area of application:

1. Rolling Stock Management.
2. Operation.
3. Traffic Management.
4. Infrastructure Management.

The scope of the project is to allow the information exchange between the different players of a domain and different areas, and process inside the domain. The target of the InteGRail is not to create a new railway system, but to support the existing ones ensuring the communication between them, detailing the specification of information sharing and the standard language and protocols to be used. In order to define and share the information semantic web technologies are used and in particular a Railway Domain Ontology has been developed. This representation can be accessed by a Service Oriented Approach.

The Railway system produces a large amount of data, but most of these are not useful for two principal reasons: the first one is that are in proprietary format; the second one is that some format are difficult to understand or

elaborate. Furthermore many data are in hand to the company who installed the system, but are not useful to it at the moment, and they are stored only for future development. Instead from the stored data a lot of useful information could be retrieved, not only for the managers of the system, but also for domain partners. All these issues could be addressed in easy way and at low cost. To achieve this scope, InteGRail produced the Railway Domain Ontology (RDO), starting from a simple area that contains the physical components and the base data concepts of the domain. This ontology is a conceptual model that could be easily interpreted and managed. The description language that was chosen for the domain representation is the OWL (Web Ontology Language); this is a W3C standard for encoding common knowledge.

The proposed solution consists of a network node that contains the semantic of the domain, and a distributed service oriented architecture with the scope to integrate and share the information described into the RDO. The ontology has to solve two main issue: the porting of the data to information in such a way that is available both for human users and computers; support the automation of analysis, communication of and extraction of complex information. In order to guarantee the prefixed targets, the existence of one standard ontology for each specific domain is essential for the InteGRail project. In this case the usage of only one ontology for the railway domain is proposed. The adopted ontology must be continuously enriched, refined and validated. Further development of the ontology guarantees that all aspects of the domain are covered and that new issues and needs are addressed, It has to be refined after each expanding activity in order to not introduce inconsistency and management problems. Finally the RDO must be one and validated to avoid semantic and conceptual errors.

In a railway domain, where there are different kinds of information sources and different types of information, an RDO ontology is the best solution to support data interchange. All the vendors that work in the railway industry applying this standard exchanging can use all data from the different systems present into the domain and using reasoning applications can `interpret` and manipulate data.

Furthermore the data stored in this way could be used for data analysis: some information are encapsulate into data, using ontology it is possible to capture such implicit information and operate on them. Data could be analyzed from a large number of different applications since the RDO is a

machine interpretable. The applications can `infer implicit information from explicit statements`.

These benefits are used during decision making: sharing between the train operator and the maintenance vendors information about the state of health of the trains could be found the best solution between trains availability and the minimization of time lost to maintenance them. The results presented by InteGRail project show that there are many advantage in using semantic techniques based on domain ontology. In particular the different ontologies developed for the different application cases of the project had good results, but the key activity is delegated to the stakeholders. It means that railway companies have to work together to unify and mantain the different domain ontologies. InteGRail is a good example for modeling railway related concepts using ontologies, but in contrast to our work in InteGRail the model is used as a data structure contract while we focus on the semantic verification of railway infrastructures.

Others works in addition to OWL for developing a domain ontology use the SWRL language for the representation of a legal railway guidelines. In [20] an OWL ontology is used, with correlation rules declared by the Semantic Web Rule Language (SWRL), another W3c standard. The SWRL clauses are antecedent and consequent pairs inserted like OWL individuals into the domain ontology. In the project the SWRL rules are integrated with railway domain adds on without prejudice consistency and decidability. The OWL ontology developed is based on the RailML SCHEMA that is an XML representation of the domain ontology, as for InteGRail, from this are developed three different schema: rolling stocks, infrastructure and timetable.

Others works are related to automated ontology extraction [4] in order to automatically grow the knowledge representation, but the growing of the ontology must be always controlled and validated because a presence of unmanageable concepts could hide errors about the railway infrastructure developed.

2.3.2 Semantic techniques

Semantic techniques for effective retrieval of information has been a relevant research activity in recent years. The exploitation of ontologies to annotate any kind of document and web sites has been proposed for building a knowledge based to be processed by intelligent reasoners which go forward

the simple search by keywords[27]. Techniques and tools for semantic annotation have been proposed for intelligent service discovery of services [2], computational resources [27], to improve context awareness [1] and for effective recommendation[3]. However most of the available information has not been annotated with explicit semantic information and keyword base search of indexed text document is still the most widespread solution. Much effort is spent for improving the retrieval efficiency in the Internet, because it is the major source of information for millions of people. Information retrieval systems traditionally rely on textual keywords to index and retrieve documents. Interpretation of "bag of words" causes losing semantics of texts[18]. Keyword-based retrieval may return inaccurate and incomplete results when different keywords are used to describe the same concept in the documents and in the queries, as the relationship between these related keywords may be semantic rather than syntactic. Concept-based retrieval methods have attempted to tackle these difficulties by using manually built thesauri, by relying on term co-occurrence data, or by extracting latent words relationships and concepts from a corpus. In [13] a method that augments keyword based text representation with concept-based features is presented for generation of new text features automatically. In fact due to the lack of labeled data, traditional feature selection methods cannot be used. This is relevant in different application contexts. In Biomedical research, retrieving documents that match an interesting query is a task performed quite frequently. In this field semantic indexing of the results of a query is presented in [22]. Relevant terms in a document emerge from a process of Named Entity Recognition that annotates occurrences of biological terms (e.g. genes or proteins) full-texts. The system is based on a learning process that starts from a set of manually classified documents. The resulting network of items implements the semantic indexing of documents and terms, allowing for enhanced navigation and visualization tools, as well as the assessment of relevance for new documents. To solve the limitations of keyword-based models, the idea of conceptual search, is addressed in [14] too. This work investigates the definition of an ontology-based IR model, oriented to the exploitation of domain Knowledge Bases to support semantic search capabilities in large document repositories, stressing on the one hand the use of fully fledged ontologies in the semantic-based perspective, and on the other hand the consideration of unstructured content as the target search space. The integrated utilization of semantic techniques and text docu-

ment processing technologies has been extended also to classification and discovery of services. In [30] web service discovery is addressed. Authors propose an ontology framework for achieving functional level service categorization, given that a vast majority of web services exist without explicit associated semantic descriptions.The proposed approach involves semantic-based service categorization and semantic enhancement of the service request.

2.3.3 Knowledge management and extraction

In [31] semantic models are used to connects various artifacts of the development process (work items, such as object models and process models), to the activities of testing. Semantic models are applied to perform online testing of the embedded systems. In particular the artifacts made in the development process are used to build semantic models for software development and maintenance. The data obtained during the early phases of the process are collected and used to obtain different types of test (g.e. test cases and test processes). The proposed methodology for testing is based on dynamic configuration models that use both semantic and syntax evolution techniques. The test are applicable for functional requirements of the embedded systems that in the early stages of development phases are described with use case diagrams and use case descriptions. In the next phase the behavior of the system, in terms of modules that can process input or generate output, is specified with state diagrams, class diagrams and use case diagrams. Furthermore the sequence diagrams are used to describe the processing of the events in the embedded systems. The software code instead represented as database that connects the classes, its methods, the attributes and a relation towards other objects represents the Event. A pattern is developed in order to precess the events to test the embedded systems. The test scenarios are described by a fixed template that contains the inputs, the outputs, the location of test and the method of testing. In the template is specified also the testing relation between all these parameters that represent the testing procedures and sequence of actions. The functional code units and the events are connected to scenarios and to pattern of testing to complete the tests.
At this point different types of test cases (integration, functional, regression) are generated and submitted to the test processes resident in the target embedded system (i.e. an evaluation boards). The generation is based on Event,

test pattern and templates which through the test method description indicates the type of test to be performed.

In various works the semantic techniques are applied to discover the relation between data involved into the application. These methods are very useful for the verification of large systems that use a big data set. Usually the relations under test could be:

- relation between input data and one output.
- relation between input data and multiple output.
- relation between input data and the configuration data of the system.

Some projects try to introduce a static analysis about the semantic relationships between data types and constant values. The relations are derived also by the semantics of each operation and operand in the code, and use them to navigate across the source code. The elements of a program has annotated in a more abstract way, for example an index in a for loop is not annotated as int or integer, but as a monotonic increasing discrete function from a lower to upper limit. Another example is a variable that is not only considered having a type and a range values, but also a set of equations that build it, g.e. control flow sensitive relationships. In such a way the run time errors are considered correctness condition, and the solution could be found using equations that tie variables together. The abstract interpretation using semantic rules address problems like efficiency, costs and time to target in the developing of reliable embedded systems. The experimentation of such method in [25] provides the following benefits:

Assurance of code reliability : the code review performed with abstract interpretation supports error detection and proves the code correctness. The debug information verify also the reliability of the code and it is not focused only on errors detection. Such methodology supports the confidence about the software reliability identifying code that could not generate a software fault. This aspect is very important for safety of applications that use embedded systems, whose failure could produce a catastrophic accidents.

Increased efficiency : abstract interpretation reduces the cost of developing the application reducing the time to find and correct run-time errors during development, when it is more simple to fix them. The abstract interpretation supports the engineer to identify the code lines that are free

from run-time errors and those that could cause a reliability breach by verification of the dynamics of application.

Reduced overhead : such static analysis reduces overhead because it is not needed to prepare and execute test cases. Time is saved also because the code does not need to be instrumented.

Simplified debugging : the time spent for debugging is reduced thanks to an easy way to identify the source of errors and not only its effects. The tracking of errors is immediate and there is not necessity to reproduce defects. This kind of analysis is also repeatable and completed because each instruction into the code is checked in relation to all its inputs.

The strength of this method is that it could be applied on already started projects and in a common development process for embedded systems without changing it; and although it is applicable only to code development, it could be very useful as start point to work up a method for log test analysis.

2.3.4 Analysis of Requirements

The Requirements Analysis is one of the most important and early phases of the process for all application types and in all models of the software life cycle. This phase is very delicate especially for safety critical systems, both for development that for verification and validation. The requirements could be divided into Functional Requirements and Non Functional requirements based on the described software characteristic . In many application domains, as the railway one, an important aspect is to identify the safety related requirements. These are transverse to the first two groups and is very important to chose correctly such set because an error could bring to catastrophic accidents. Many times, in order to cover all possible safety requirements, and to avoid to miss controls on reliable functions or aspects, engineers decide to consider all the set of requirements as safety related, increasing considerably the effort to validate the systems. The requirements could be divided in more groups as for example the performance, design, customer requirements and so on, however for our scope such division is not important.

The requirement analysis is the process to determinate user expectation regarding a software or a system and is like a contract between the customer and the company. The requirements must be actionable, measurable, testable

and detailed. In order to perform a good requirements analysis the following activities should be performed:

- *Eliciting requirements*: this phase is performed to obtain the requirements from all stakeholders involved into the project, as for example customers and the future users of the software/system. People that interact in different way with the system helps to consider different views and discover the various issues.
- *Analyzing requirements*: after that all requirements are written, they must be reviewed and corrected despite specification unclear, incomplete, ambiguous and contradictory.
- *Recording requirements*: requirements have to be documented, and this could be done in various forms, such as natural language, UML diagrams, formal specification and so on.

In the requirements analysis many problems came from method to write down them, often it is used natural language that could be interpreted in different way based on the experience of the people involved. For example an engine driver could use the terms signal independently for the lights signals and the boards, but a system engineer may not understand the difference and so consider them the same object. At the state of the art there are different approaches to address such issues, some of them are alternatively or complementary. Some works, especially related to aerospace and defense, use formal methods for the requirements analysis in order to prove the absence of errors. Another approach is to use a Model Driven Architecture (MDA) to produce a system based on models. In MDA a modeling language is used to define the rules to follow to design the system and describe its through models. The system requirements in this way are described in a well-defined language. Two modeling construct are used in the MDA approach: the Platform Independent Model (PIM) and the Platform Specific Model (PSM). From the first abstract model PIM is derived the PSM. Note that from one PIM could be obtained different PSM, improving the reusability. Even if the formalism used in MDA resolves many of the problems related to requirements specifications, the correctness of the PIM depends from the information retrieval, the flow of the real world system and the work of systems' analyst.
In the last decade, the use of semantic techniques is growing, also thanks to the support of the research about the semantic web and the web services. In

[19] is presented a semantic approach to requirements specification based on ontology domain. The problems related to the use of natural language into system specification is addressed using a reasoner for the text analysis. The use of such solutions has the aim to not have a great impact on the applied process. Others methods, as the ones mentioned above, are not considered to avoid the duplication of the activities that could produce just a usefulness addition of new artifacts to those already available. The results is a better quality, but spending more time and resources.

The application of a semantic approach aims at supporting the user to address clarity and content related issues. The problems related to clarity aspect, which are analyzed in the proposed approach, are the *use of problematic phrases*, the *terminological inconsistency* and *missing contextual information*. The sentences are processed checking both the constructs and the used words. The use of different vocables into requirements to annotate the same concepts is avoided by checking if all the concepts present into the specification belong to a defined glossary. The use of problematic phrases and the missing of contextual information is addressed by a logical analysis of the period. Furthermore to help the user to discover the miss of interaction between the components of the system, a set documents and diagrams are produced that shows the right connections.

The approach is based on the use two types of glossary: the *entity glossary* and the *action glossary*. The first one is a vocabulary where each entity is tagged with its own categories. For example in a railway domain the entity RBC will be annotated as subsystem and the entity driver is annotated as user. The glossary helps also in the analysis of the sentences because for each object is specified if it can execute an action (agent) or have to undergo it (passive). All the actions that can be specified for the agents in the specification are listed and form the second glossary called *action glossary*. In order to simplify the analysis of the requirements the use of the natural language is conditioned to the use of some pre-configured sentences. Also the sentences are tagged with the type of requirements for which can be used, g.e. the phrases are divided into a set to express a capability, a set to define constraints and so on. The requirements written in natural language, following the rules described, are analyzed syntactically and prepared for the semantic analysis. The phrase is matched with the predefined structure, and based on this the sentence is converted into a series of tokens; after is derived a

syntax state machine of the requirements and can be checked if it is syntactically correct. Finally upon this result is performed the semantic analysis. The semantic analysis is used to find problems into requirements as dependence and incoherence between requirements and missing requirements. The semantic analysis is based on domain ontology for knowledge representation. The specification, structured into the syntax state machine, is transformed into a semantic graph using the domain ontology, the glossary and a set a rules developed ad hoc. Each requirement is represented as an instance in this core requirements ontology that models the specification documents and the related artifacts. The semantic representation of the requirements specification at this point is inferred and it is used for reasoning with SPARQL queries, taking as true reference the domain ontology. This approach use two ontologies, one developed from the requirements and one, developed ad hoc, that represents the domain knowledge. As could be easy to understand the domain ontology is considered corrected and is used to check if the semantic representation of the requirements violets the first one. The problems about this method is the correctness of the domain ontology, that must be validated, maintained and updated. The limit could be found in the definition of the glossary that have to be performed and verified for each project, and an error in this phase could produce wrong syntactic and/or semantic models. Instead in [37] the traditional requirements engineering are integrated with semantic techniques based on ontologies. The ontologies are used into Requirements Analysis in order to:

- define the requirements model;
- define acquisition structures for domain knowledge;
- describe the knowledge of the application domain.

The discussed approach uses three different types of ontology to perform different activities: *Requirements Ontology*, *Requirements Specification Document Ontology* and *Application Domain Ontology*. The requirements specification are annotated with RDF triples that represent the conceptual relation between an entity of the requirement and a concept or a set of concepts of the ontologies. The requirements ontology is an abstract representation of the conceptual requirements furnished by the customer. This ontology is developed to address issues regarding the elicitation process reducing ambiguity and incompleteness in requirements specification. In this phase it is simpler

to define constraints on requirements and to support the verification and validation phases. The requirements specification ontology is used to support the requirements analysis. The semantic reasoning based on the ontology can reduce the incompleteness of requirements specification. The independence of the structure of the requirements described into the ontology and the contents of the specification documents help the definition of reusable structures applicable to different projects. Finally the application domain ontology is the way to represent the knowledge about the specific topic. Such information are necessary to understand the application goals. This ontology contains the semantic representation about the system requirements that the user is developing. Independently from the techniques implemented to perform semantic reasoning the division of the ontology according to three different domain is very interesting. As asserted by the authors this approach could have advantage in the reuse of the ontology, the modeling of the specific ontology for the particular task and a easy maintenance of the knowledge base.

We consider an ontology as a semantic domain to provide the meaning for requirements and discuss the potential of the RE techniques using an ontology as a semantic basis. More concretely, we adopt the technique of denotational semantics and consider mappings from artifacts (incl. requirements) to the ontology. An ontology consists of a thesaurus and inference rules on it, and the thesaurus includes the words and their relationships. Each word in the thesaurus is frequently used in a certain subject domain and it denotes an atomic semantic element that has a unique meaning in the domain. We can map artifacts to the words in the thesaurus and the meaning is provided for the artifacts by the mapped words. As a result, we can reason about semantic properties of the artifacts by using the inference rules on the words. This is a basic idea in this chapter. As a result, we can have a light-weight semantic processing of artifacts for assisting RE activities, e.g. semantic consistency checking of requirements, retrieving the requirements that are semantically similar as reusable components, etc. Our technique is inspired by the technique of Semantic Web, where HTML texts are annotated with semantic tags derived from ontological components. In Semantic Web, the meaning of information on the web is defined using the annotated semantic tags so as to make it possible to analyze and process the web contents by computers. The semantic tags represent the meaning of information where they are annotated. In our technique, any kind of artifacts related to RE process, including spec-

ified requirements with natural language sentences or UML diagrams, meta models, etc., are annotated with (mapped to) semantic tags (ontological components) so that the existing RE techniques can semantically process them by computer. Thus we can use the word Semantic Requirements Engineering by the similarity to the idea of Semantic Web.

2.3.5 Traceability through Semantic Research

The traceability between requirements and test scenarios is an activity very onerous to perform and to maintain over time. There are some approaches, based on semantic techniques, to retrieve automatically such information. One approach is defined in [21] where an automatic reconstruction process is applied using Information Retrieval (IR) techniques. In order to discovery the links between the requirements and the test scenarios the Latent Semantic Indexing (LSI) is used. The assumption to be done in order to have success using the LSI is that the documents should have a latent structure to retrieve information and make the links. Usually in the process a great attention is given to the traceability between requirements at different level especially in the development phases, e.g. the connection between the system specifications and the software requirements specification. However in the verification and validation activities is important to have the Matrix Traceability in order to verify the following activities:

Test case coverage : the implementation of the test case have to guarantee that all functional requirements are covered into the verification activities. Usually the test set is planned in the design phase and shows how well the specification are covered. The aim of this phase is to reach the 100% of coverage of the functionality.

Test pass coverage : the test pass coverage is the set of requirements for which the tests performed are all passed. This set of information are used by the company to show to the customer which are the functionality that could be considered correct and ready for the acceptance test.

Acceptance coverage : this is the set of tests performed together with the customer to give the evidence about which are the functionality and related requirements ready for the release.

Evaluation coverage : it indicates the requirements that could be reused in the actual project or in the next one.

The LSI is a technique that uses a Vector Space Model and it is based on the assumption that there is always an hidden structure of words usage in the documents and it can be discovered by statistical approach. The assumption is linked to the classical issue as synonymy and polysemy. The latent structure is discovered using the terms, the documents itself and the queries developed by the user. This allow to overcome problems like the independence of the words and the issues connected to synonymy. Furthermore in such a way is not needed to define before the structures used into the documents, a categorization of the words and more in general to not produce extra information for the elaboration, reducing the effort to make the traceability.

The first step in the application of this approach is to identify the traceability model to retrieve the correct links between the tests and the documentation. The model is very important because indicates which kind of links are admissible for the project and drives all the remaining process. Indicate a particular type of links rather than another bring to very different results. After identified the relevant concepts the documents have to be arranged in a form that is suitable for the automatic processing. This task is manually performed and its difficult is related to the heterogeneity of the artifacts and the structures defined (i.e. if are used templates). The prepared text is used to simply perform the IR analysis like word elimination and indexing. At this point, using as input a constructed term by document matrix with LSI technique, a similarity matrix that contains a distance relation between all elements is produced. The matrix contains all the possible recovered connections between the elements of the artifacts. At the end must be decided which of the similarity have to be considered as link. This is performed choosing two threshold, one is constant and defines the minimum bound to have a link, if no similarities have a value greater of the constant no link are produced. Instead the variable threshold is used to define among the candidates grater than the minimum which are the correct links. Finally the links identified are used to produce the coverage.

In [33] an automatic traceability recovery is used to support the maintenance activity. The traceability between different types of documents produced into different phases of the development process is very important to understand the relation between the software artifacts and the various system components. The main issues related to a good traceability are the use of different languages into the documents (e.g. Natural language, modeling

language, programming language), and the different levels of abstraction to describe the system in each phase of the process. Also if many tools, like Doors, are used to support the traceability between the documents, in company's processes the maintenance of the traceability matrix is often not an activity performed as well as it could be done. Some approach use Information Retrieval (IR) techniques in order to implement an automatic traceability between various types of artifacts, such as code upon system requirements and test cases. However in such methods there is the lack of semantic relations between artifacts, along horizontal and vertical direction. Introducing a semantic approach for the traceability recovery could improve the quality and the efficiency in such phase.
The approach used in [33] implements the integration between structural and semantic information, building an ontology description for each type of artifacts. In particular text documentation written in natural language and source code are considered. The classical IR techniques are replaced by Text Mining (TM) on natural language documents and a source code parsers for the software implementation. The concepts and the instances present into the two ontologies are used to recognize links between requirements and source code, but also to perform semantic reasoning. In the figure 2.1 the interaction between the user and the two ontology automatically developed is presented.

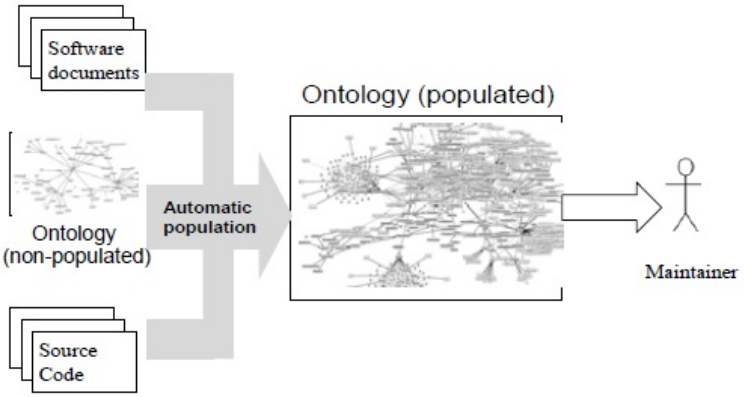

Fig. 2.1. Traceability recovery with ontology approach

Use of semantic methods to analyze the relations between different types of artifacts helps to the user to retrieve more information, not considered at

the state of art by the available tools. The usage of ontological representation allows the use of the Natural Language Processing (NPL) to discover hidden links between concepts and sections of the documentation. The exploitation of an ontology for the different type of artifacts, as for example system requirements written in natural language, code and test script, allows to use and highlight the same concepts from different resource and integrate them easily. Using the proposed approach the following problems related to traceability are addressed.

Semantic Links : definition of a meta model for information and knowledge representation. The model must guarantee the correct representation of the information, an unified representation for all the artifacts, clarity of the representation. Such model should be independent from the application domain in order to be reusable for other domains.

Traceability Links : implementation of traceability method must be accurate as performed by human, but should have best performance in terms of time and costs. The proposed method is based on natural language processing, but is impossible to fully replace the human work because by NLP it is not possible to guarantee the correctness. However the traceability links would be obtained in a semi-automatic manner and more quickly than the manual one. The links must be well defined and the method should be not intrusive with the current process.

Textual artifacts : the documents and in general all types of artifacts used in a project come from many source and vendors and so have different formats and data representation. The application must retrieve the information and perform semantic operation also with such not structural documentation. Another approach could be to arrange the documents in a standard format, but this could be very expensive.

Integration : the application have to be integrated into a tool already provided by the company. The user does not want to change its process or the set of tools used.

Experiments : the infrastructure have to support experimentation and should be tested in different domains.

We will concentrate only on the first three features, which are those of major interest and more strictly related to the methodology that we will describe later in the nect Chapter.

The chosen knowledge representation language is the OWL DL, where DL stands for description logic. The information are described with concepts, relation and individual and inferred by the ontology reasoner Racer. There are two sub ontologies defined, the first is the *Source Code Ontology* and the second is the *Documentation Ontology*. The source code ontology contains all the information related to the code and is dependent from the programming language used. The concepts of the source code are in relationship toward conceptual links that explain the roles in the program, i.e. could be present a relation read between a method and a variable showing that a read instruction of the variable is present in the body of the method.
The documentation ontology is the domain knowledge base and it is used to retrieve and to recognize all the concepts present in the software documentation. In addition there are also others aspects present in the ontology related to some constructs typical of the artifact used and of the domain. These information are the structure of the documents in terms of paragraph, sentences and text position, lexical information as keywords and patterns name and relation between classes (also relation relating to the source code ontology). Finally the Documentation ontology is automatically populated with code entities through a text mining system.
The problems related to traceability links and textual artifacts are addressed using semantic techniques on the two ontology described before. In a first phase the two ontology are populated. The source code is tokenized and a syntax analysis is performed in order to produce an Abstract Syntax Tree (AST). The AST is analyzed and are identified the concepts and the relations among them, these are finally used to populate the ontology by an OWL Generator. On the other hand is populated also the Documentation Ontology, following an approach more complicated due to the unstructured format of the high level artifacts. A text mining system have been developed ad hoc in order to extract knowledge both from specification documents and source code. The phrases are splitting, analyzed and tokenized. The entities recognized as proper of the domain are tagged (The core ontology to populate already exists) with the right concept. For complex named entities are used grammar rules in order to refine it; g.e. if it is recognized "'the ClassName have"', only the entity *ClassName* have to be be instance of the concept *Class*. So the relations among identified entities are discovered through syntactic analysis and grammar rules. Finally the core domain ontology is populated with the

extracted knowledge. Now the links between the two ontologies can be find in a relative easy way. There are many techniques that can be used to solve the problems of the alignment between the two sub ontologies, furthermore this issue is simpler because the document ontology and the source code ontology have many concepts in commons. After the link identification between code elements and the requirements entities the user can run ontological queries about the properties of the software and retrieve information (g.e. description) regarding the Class under inspection. The query could be used also for instantiate others links not found during the alignment and retrieve in such a way more information.
The use of such approach is not limited only to retrieve links between code and specifications, but also to classify the different sections of the documents according to its semantic contents and links different levels of abstraction of the same topic. For example, the documents type *Metrics Software* could be all the paragraphs that contains the concept of a metric.

2.4 Final Considerations

Applying the proposed methodologies discussed in this chapter the railway domain could obtain benefits in terms of time to market a cost since the first target is to accelerate some repetitive and tedious activities. The test reuse and the enhancement about coverage of requirements assure that no issues are overlooked since the first phase of development process and the probability of finding errors in the last phase or in deployment phase are minimized. In addition the application of semantic techniques allows the user to simply analyze data and retrieve hidden information from them.

However there are many issues to take into account to widely integrate such activities into a industrial safety process. First of all the companies are wary to changes their process to adopt new methodologies. If a process works, let it going on. In particular the critical aspect is to replace a running technique with one not yet used in the company. The problem is that introducing a new activity in a well defined work breakdown structure brings some modifications to the overall process and in particular to the relation between the new activities, the work packages already present and the input/output between the different stakeholders. Therefore in order to have success in apply innovative methodologies, they should be integrated into the current safety

critical process in a minimally invasive way and trying to replace gradually the existing ones.

Beyond the language to be used, that as described in the previous paragraphs seems to be OWL, there are a lot of problems related to structure that the ontologies have to assume. A particular concept could be viewed by a company as a relation and by another company as a family of entities. For what regards to ontology or others type of knowledge representation maintenance the key aspect is that to have a good result the model must be unique and always updated. In such way avoiding problems like misunderstanding and exchange of information is made easy. An auspicious solution is that a consortium, like the UNISIG for ERTMS/ETCS, will be created in order to develop and maintain the knowledge base. Finally a big hurdle to overcome is the companies reluctance to share information with others vendors. There is a thin bound between what is the sharing of sensitive information and statistic data and the target is the identification of the right details to share without breaking patents and technical solutions. A company is more worried about which advantages can take the competitors knowing his data that getting a common profit from sharing information. This scare imposes limit collaboration and hinder the progress. The only solution that could stimulate the company to accept such methodologies as well as having the security about its intellectual property is a strong demonstration about the economic benefits at stake.

3

A Methodological Support for Testing Safety Critical Systems

Here we introduce new methods to support and improve the human activities in the safety life-cycle and to increase the level of automation of repetitive and tedious tasks.

We aim at reducing the time spent without loss in quality by the exploitation of semantic techniquesm, reducing also the error probability.

3.1 CENELEC life-cycle

CENELEC is the European Committee for Electrotechnical Standardization and is responsible for standardization in the electrotechnical engineering field. Its center is located in Brussels. It was founded in 1973 as unions of two previous European organizations: CENELCOM and CENEL. The CENELEC sub-sections EN 50126 [9], EN 50128 [10] and EN 50129 [41] address the development life-cycle of railway systems. EN 50126 describes the specification and demonstration of Reliability, Availability, Maintainability and Safety (RAMS) for the railway application. EN 50128 and EN 50129 specify the necessary activities to develop, verify and validate respectively the software for railway control and protection system and the safety related to electronic components for signalling.

The CENELEC life cycle for development of railway systems is presented in Figure 3.1.

The normative describes how to apply the verification and validation activities during all the phases of the system life cycle and in which ways the V&V RAMS contributes to the overall system assurance V&V. In the follow-

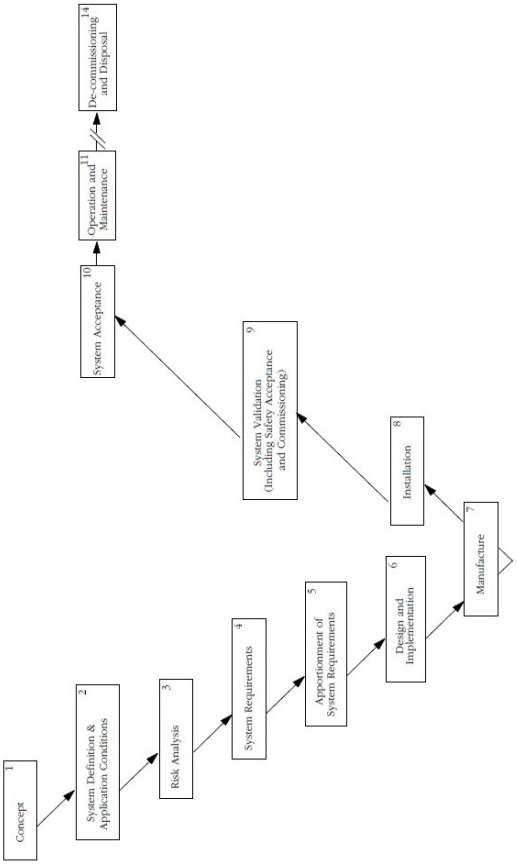

Fig. 3.1. Vmodel of the CENELEC life cycle

ing we summarize only the main task for each phase, to analyze deeply all the activity performed in each task we refer to the normative [9].

Concept : the idea of the system is developed. Usually this phase involves the supplier of the service, and not to the vendor of the system.

System Definition and Application Condition : the mission of the system is defined, together with its boundaries and the application conditions that can influence the service. In this phase the V&V plan and the safety plan are set up in a preliminary way.

Risk Analysis : the events related to the dangerous situations are identified in this phase, and it is declared how they are managed. The risk analysis

is not performed only at this time, but it is repeated periodically during all life cycle.

System Requirements : the system requirements and in particular the RAMS requirements are specified and analyzed. The V&V plan is detailed in this phase.

Apportionment of System Requirements : for each subsystem or component the requirements are apportioned , and it is defined also which is the level of safety and RAMS in general that the subsystem or the component have to satisfy.

Design and Implementation : the system is designed and implemented, at this point evidence that the system is compliant to the specifications should be given.

Manufacturing : the product is finally developed. During this phase all the techniques to guarantee the correctness of RAMS process must be applied .

Installation : the components are installed and integrated in the field. Ahe alignment of the system as build to the design has to be checked.

System Validation : the system is validated, all data are collected to give evidence that the system is compliant to requirements and in particular to the RAMS elements.

System Acceptance : at this step an evaluation of the compliance of the system , and of all subsystems to the acceptance criteria to entry into service are performed.

Operation and Maintenance : the activities for reducing the risk and maintainin the compliance to RAMS system requirements must be performed .

Performance Monitoring : the system performances have to be checked periodically to maintain confidence to the RASM requirements.

Modification and Retrofit : all modification improved upon the system must be performed in according to the CENELEC normative.

Decommissioning and Disposal : finally also the decommissioning and disposal is completed according to planned activities and to safety criteria, which are linked also to environment where the system is installed. If a decommissioning regards only a part of the overall system, it must be assured that the RAMS criteria are respected for the interested external facilities .

3.2 New Methods in the CENELEC life-cycle

We introduce new methods in the CINELEC developing process.

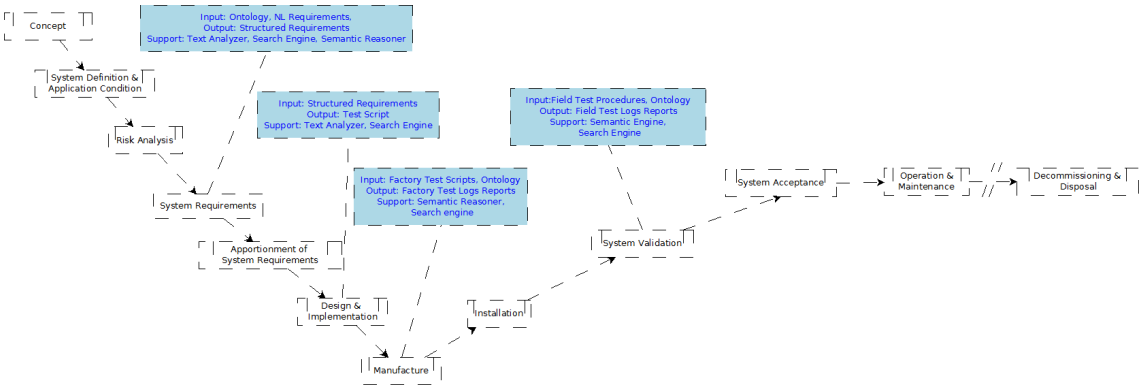

Fig. 3.2. The V representation in the cenelec life cycle

In Figure 3.2 we highlight the phases of the CENELEC life-cycle we address by introducing new techniques and technologies.

In particular the proposed approach covers the V&V activities in the fourth, the sixth, the seventh and the ninth phases. The apportionment of the requirements(5) could also be supported, but since our case of study works only at system level, the activities related to this phase are missing.

In particular they are applyed in order to complete the specific tasks in this way:

Phase 4 *System Requirements*: In this phase the requirements in natural language are translated in a structured format with the concepts of an ontology, using text analyzer and search engine. The requirements are refined using the support of a semantic engine. The output of this phase is the structured refined requirements that are used in phase 6 to generate the test script.

Phase 6 *Design and Implementation*: the structured requirements are the input for the generation of the test scripts. The supporting tool to automatize this activity uses a text analyzer and a search engine.

Phase 7 *Manufacture*: the test scripts produced in the previous phase are executed in the simulation environment and the resulting text logs, to-

gether with the ontology, represent the inputs for the analysis, which uses a semantic reasoner a search engine for information retrieval.

Phase 9 *System Validation*: like for the previous phase the analysis of the test logs is supported by a semantic reasoner that uses the domain ontology and by a search engine. The difference with the previous phase is that test scripts do not exist, but the procedures are executed directly in field on the real system.

3.2.1 Test Definition

Starting from the stable specification of requirements, the tests have to be planned and designed in order to cover the functionalities of the system. The developed test cases must to cover all the testable requirements, and at least in nominal and in degraded scenarios. The time is spent for both planning and translating the test scenarios into test scripts manually. The aim of the innovation of to the test definition phase is the improvement of the quality of the test cases, avoiding errors in their specifications and consequently a new execution of the test itself. Another goal to achieve in this phase is the reduction of the time to write the test scripts from the test case, reducing the overall effort. In order to reach this target we introduce semantic techniques to support documents analysis and information retrieval and a method to derive in automatic part of the test case from the requirements. In order to support this activities a new task was introduced to translate the specification in a more usable form.

The original phase of the verification and validation process is divided into three sub activities:

Requirements transformation: Here we aim at reducing the ambiguity of requirements which are originally written in natural language. As shown in Figure 3.3 the requirements written in natural language are translated into a semi structured formalism by the support of semantic techniques. This method helps the user to be warned about errors such lack of consistency, incoherence, duplication during the requirements analysis. The used formalism is always the same. In particular the use of semantic techniques allows the user to highlights requirements that are in conflict with the representation of the application domain. The use of a structured language helps also the automation of the following activities.

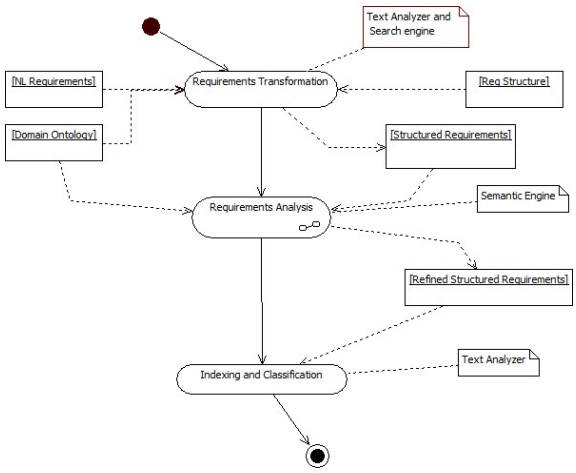

Fig. 3.3. Test Definition: requirements transformation

Indexing and classification: The objective here is to support the developer for fast and effective retrieval of requirements by visual interface and text query. The possibility to look for related requirements allows for better management of the documentation and for tracing in exact and quick way the relations among the data. The user has always under control all knowledge that he needs without losing some relevant information that are not in the expected document. This activity is useful also to check if the changes to a requirements is propagate to all artifacts semantically related, to verify the requirements traceability matrix and the test coverage. We aim at processing documents using both text based processing and the semantic techniques.

This activity, shown in Figure 3.4, is performed in automatic way. It will exploit the structure of requirements expressed by a semi-formal language, natural language description and other metadata.

Automatic test case generation: since the time to write down the script for the testing is expensive and the structured requirements are usable for automatic processing, the functional test could be retrieved directly from the specification. In particular starting from the specifications the pre-condition and the post-condition can be obtained and can be translated into the proprietary format for the execution in the simulation environment. Even if the effort to develop a system for automatic test generation

can be relevant,the expbected benefit motivate this activity since the test script production is one of the more heavy task. The core of the transformation activity is based in a set of transformation templates and rules in order to produce different kinds of test.

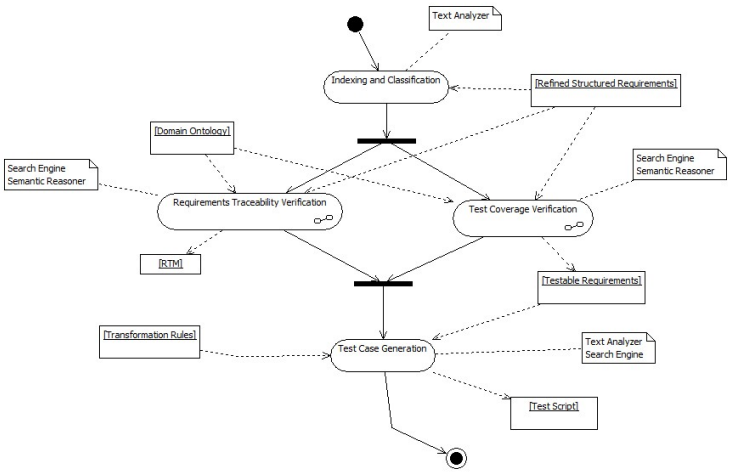

Fig. 3.4. Test Definition: indexing, classification and automatic test case generation

3.2.2 Test execution

This activity is substantially the same, also if the simulation systems evolve continuously. About this subject a lot of research are performed to parallelize the test execution and to recover the system in case of errors without the attendance of the user. For example distributed environments are enriched with self checking systems that when the number of failed test is above a fixed threshold, statistically determinated a priori, the system is restarted and tests are executed again.

3.2.3 Test report analysis

This phase is manually performed. When a test fails the user has to understand the error that brings to the failure and must give indication to support the issue fix. This activity is most complex and the hardest one. The user has

to know very deeply the system and needs to manage a many variables and from the values of those variables has to rebuild the state of the system and its history. The aim of the innovation is to improve the quality of the analysis and reduce the effort spent supporting the discovery of similarities between errors and test log failed and the highlights of hidden relations among variables states.

In order to support the activity of log inspection and analysis this phase is divided as reported in Figure 3.5:

Indexing and classification of test log: the results of the test execution are some test logs produced by the simulation environment. This test logs have to be categorized and stored in order to be managed and retrieved quickly when needed. In addition a good categorization is important to set up a semantic support to the analysis.

Log analysis support: the log analysis is supported by semantic techniques. This activity, in conjunction with the indexation of the documents allows the user to find the cause of the errors earlyier thanks to retrieval of hidden information and to the estimation of similarity between different cases.

3.2.4 Test Report document Drawing Up

The verification and validation report is drawn by the user, it consists of a set of documents containing the results of the test activities and the justification of test failures. In this phase the criticalities are the management of the reports of the testing activity and test coverage evidence. Automation and formalization of the output of the previous step help to have all information available in a standard and understandable format and to speed up the whole activity.

3.2.5 The new effort distribution

The changes to the break down of the work packages for the verification and validation activities aim at improving the efficiency of the overall process and help to have a better quality moving the effort from the repetitive tasks to the initial phases, which are the more thorny and difficult.

In Table 3.1 it is shown how the phases of the process are splitted and enriched. The overhead due the introduced activities indicates the effort spent related to the 100% and not to the amount of the percentage of effort actually

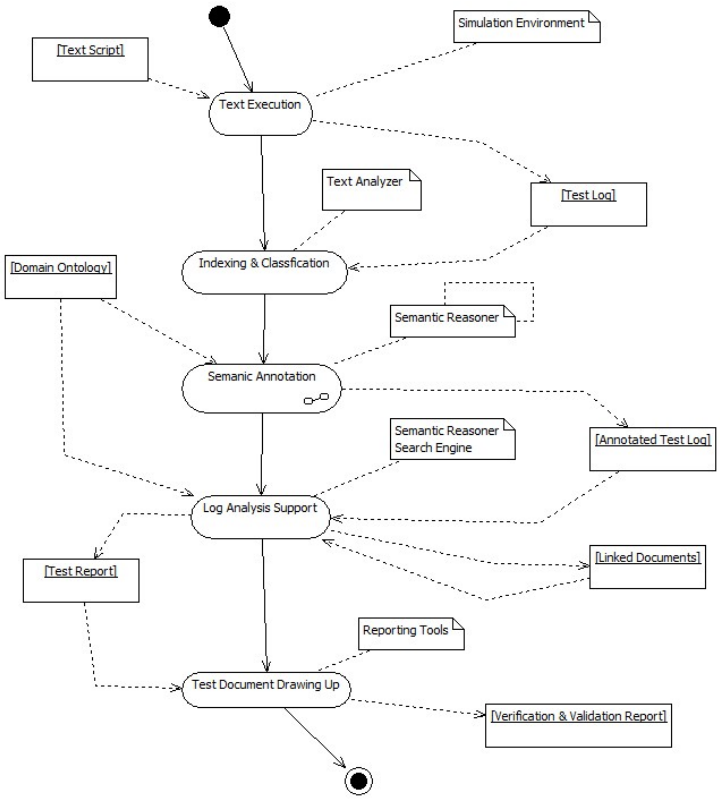

Fig. 3.5. Test Definition: indexing, classification and automatic test case generation

spent, e.g. the sum of the activities *Requirements transformation*, *Indexing and Classification* and *Automatic test case generation* is not equals to 25%, but is grater since the requirements transformation introduce more time consumption in early phase. On the other hand the overall Test Report Analysis that before consisted of the 50% of the effort, now is only 35%. However since the total is always 100% it is not possible from this table understand if the modification made effectively lead to an advantage. In chapter 5 we will present the obtained results.

Actual Activity	% of effort actually spent	Proposed Activity	% of effort hypothesized
Test Definition	25%	Requirements transformation	22%
		Indexing and classification	10%
		Automatic test case generation	8%
Test Execution	15%		15%
Test Report Analysis	50%	Indexing and classification of test log	8%
		Log analysis support	27%
Test Report document Drawing Up	10%		10%

Table 3.1. Effort spent in the in each activities of the test process

3.3 Techniques and Artifacts

In the following paragraphs we describe the artifacts and the techniques that will be used to innovate the phase of interest of the safety life-cycle.

3.3.1 Ontology

Semantic techniques are based on the use of a domain ontology for the knowledge representation. The ontology has the property to abstract the knowledge from the symbol level representation used in a contest or by an agent. It is a way to formally describe the knowledge through concepts that are supposed to exist in a specific area of interest. The world and the experience that a user has of that domain are abstracted and simplified in order to be used for different purpose. The essentials elements composing an ontology are the Classes, the Individuals and the Relations:

- The classes: are the fundamental concepts of the domain (g.e. motorbike in the domain of engine vehicles), they are collections of individuals.
- The individuals: are members of a class; they can inherit all or a part of the attributes of the class (i.e. properties and restrictions).

- The relations: are binary relations between two entities. They can be of different types and can have also some attributes that give information about the relation characteristics. In some representations the relations could have for example disjunction compering to another class and restrictions about values.

The ontologies are a powerful methods to transfer knowledge about a domain and they are used in different applications as semantic web and artificial intelligence. However there are some issues to address. First of all in the development of the ontology the computational properties for buildig reasoning systems must be considered. Some representations have a high abstract level and provide features which may useful but do not guarantee a decidable reasoning procedure. Introducing some constraints on the ontology formalism, and limiting in this way the expressive power, maktes it is possible to obtain through a reasoner certain computational claims, g.e. consistency and complexity. Another issue about the ontology is the boundaries of the representation and the ontology growing. These two aspects have to be managed to not introduce errors or to not trespass to others domain, modifying the point of view of the world and having a bad impact on the representation.

The use of ontology could support the description of requirements. A formal representation of the application domain can help to identify when the specifications are in contrast with it, reducing the problems of lack of clarity, incoherence and so on. Users can collect their experience and producing machine readable soruce of ingormation and can help to extend the process activities that can be automated.

3.3.2 Formal language for requirements

Even if people speak the same language, but meanings given to concepts can be different, because they comes from the personal experience. Fr this reason use of natural language to describe requirements leads to many problems as ambiguity, inconsistency and misinterpretation of specification. These problems entail the late discovery of errors during the system testing and then imply more work to recover them.

The main features that cause the use of natural language is that in this way the information could be shared among stakeholders with different competences and belonging to many application domains. Speaking about a problem

of a particular application domain using formal language could be impossible among some kind of people: the expert of railway systems and the expert of formal modeling may not understan each other. The rail engineer can use the terms from the domain context, but not the formal expression and the logic used by the expert of modeling.

Therefore we want to introduce a semi formal notation in order to represent the requirements in a way that avoid the main issues of the natural language. The availability of such formal description for requirements allows us for a more effective text analysis and indexation of documents. At the same time the semi-formal language and a domain ontology allow for semantic classification and discovery of the requirements. Therefore the system requirements are transformed into a semi-formal notation, with a well defined structure, grammar and semantic. The first element that characterizes the semi-structured language is the definition of a schema specification to identify the structures that are allowed to construct the requirements. For example not all the sentence types are allowed. The user can use only active phrases, with always the same format (g.e. subject+active verb+complement). The elements of the phrase are tagged, usually by a markup language, to indicate the type of information contained.

In this way the requirements are translated into semi formal language through a finished tag vocabulary that could be used according to preconfigured structures. The vocabulary is divided into two main set of tags: the resources and the empty structures. The resources are the objects of the application domain and the requirements describe their functions and properties. Resources of this language are the main class entities of an ontology or also words of a thesaurus. Function and properties are semantic object properties and data properties. Empty structures are connected using logic operators (or, and, then) in order to construct the requirements in the form described in the schema (usually an XML schema). Once the structured is build it could be filled with the words from the knowledge base. Usually a set of constraints in the words to use are set; for example since the empty elements are tagged, they can be filled only with words that are instance of the same class.

This kind of representation of the requirements in addition give a more readable and clear format to each sentence of the specification, with all the

following advantages, and it allows to perform automatic activities and to simplify the use of semantic techniques.

3.3.3 Automatic test case generation

The test case generation is a tedious work, which, when performed manually, requests a big effort to translate the test case specification into the test scripts. Furthermore this activity is not free from errors, especially in the connection to the requirements as a link between what are the thesis to set up and the constraints to satisfy. The actions to be taken, in order to improve the above mentioned phases, are illustrated below. The generation of test in automatic or semi automatic way are supported from the specification of the requirements in a formal or semi formal way. Moreover, the test generator has to define rules to customize set of tests, if needed.

To be noted that the test execution is independent from the approach and the technical innovations proposed could be applied to different subsystems from heterogeneous providers. The first step for the automatic test case generation is to translate the system specification requirements (SRS) in a formal requirements specification language (RSL). It could be a formal method specification as algebraic approach where the system is described in terms of operations and their relationships, or model based approach like *Petri Nets* and *Z language*, o semi formal methods as described in the previous paragraph. In the definition of the test can be used two different approach, that could be also complementary.

1. Automatic test specification from the SRS: for each system requirement a test should be automatically specified, in order to determine whether the system satisfies the requirement.
2. Customizable test generation: it should be possible to define rules, which allow the user to generate a customized set of tests.

The test oracle should be automatically defined: it should define the expected outputs, in order to detect if the actual outputs are correct or not. Usually the executable test scripts are expressed in a proprietary format, but to improve the scalability and the compatibility with other domains an intermediate step is usually introduced to represent the test script in a standard format.

Moreover the test results should be automatically analyzed, comparing the actual outputs with the expected ones, defined by the oracle. This activity is not common to all domains, but depends on the particular application and on the specific test environment, so it is specific for each company. Usually a test generator is composed by the following components:

- Test Data Generator: generates the test case from Requirements Analysis.
- Oracle: calculates the value of the expected output.
- File comparator: compares the outputs of the test execution with the Oracle prediction.
- Report Generator: implements test traceability and results of test campaign (i.e. log error, test report).

This kind of methods produce a general test scripts. It is necessary to specialize the script for the actual simulation environment to execute the test. In others words the test scripts produced by such automatic systems are something like the abstract test described above, but a little bit general, because also the particular commands and instructions are not specified.

3.3.4 Text processing

Text processing is the basis of the information categorization and information retrieval from different types of sources. First of all text processing allows to find text strings or sub strings into a document or in more documents. In such a way the text processing is used when we want to find a word in a document using a simple editor like the *find* command in OpenOffice by Oracle. This is the same command that we execute when we search something by Google, but with a powerful algorithm and the use of supporting services.

In our work text processing is used to retrieve information and to perform operations that can help the semantic inference. In particular the text processing can be performed with complex engine that support:

- search algorithm;
- ranking documents in relation to a particular search query;
- file indexation and the file locking for concurrent index modifications.

The quality of the text engines is realted to the characteristic of such services, in terms of performance (accurancy and efficiency of the algorithm) and how the operation is performed (indexing and searching in mutual exclusion,

indexing and searching in parallel, type of calculation for the ranking and so on). Furthermore common text processing engines export others services or APIs to implement them on their own.

The indexing is a process of converting text data into a format that facilitates rapid searching. A simple analogy is an index you would find at the end of a book: that index points to the location of topics that appear in the book. A typical method to stors the input data in a data structure is called an inverted index, which is stored on the file system or memory as a set of index files. Most Web search engines use an inverted index. It lets users to perform fast keyword look-ups and to find the documents that match a given query. Before the text data is added to the index, it is processed by an analyzer. The process of analysis converts the data into documents as rich text in atomic units of searching called terms. In order to achieve this task the phrase must be simplified through techniques of grammar analysis. A set of operation should be performed in order to extract the words, delete the common one, and reduce the words to root form. For example if from sentence is extracted the word "classes" the respective term is the word class. The subject analysis of a document is performed assigning the document with terms elicited in the linguistic statement, where these terms can be translated to conform with the terminology of a controlled vocabulary of indexing terms, for instance according to a thesaurus or a classification scheme. This kind of operation about forms are related to the filter that could be applied to the indexing process. The analysis operation as the removal of words or the reduction of the form has a positive effect in the reducing the index size and improve the performance of the search, but on the other hand there is a negative effect on the precision of the query. After the indexing the other basic operation, as mentioned above, is the searching of indexed data. The Search method returns an ordered collection of documents ranked by computed scores. Furthermore these are not the only capabilities of the text processing, others features as the ranking of the document, the differences in terms of contents, and the analysis of user queries in order to furnish better results are others service that can be found in the most common software. This kind of support helps the user to retrieve information from one or more documents, and to update them in a very limited time, comparing the same task performed in an manual way.

3.3.5 Semantic reasoning

Tests must cover the functional specification of the system, they are based on and must be compliant with the related normative. Test descriptions provide a set of behaviors of the system, a representation of the state of entities, signals and anomalies. They also include checks to be verified in order to declare the test passed, also in terms of not functional requirements. The test plan gives the evidence that all safety and testability requirements are covered, and specifies which test set covers each functionality. The complexity of such mapping is due to the fact that a test description for a specific function could cover also other features and the related requirements. In this scenarios automatic discovery service would be useful to look for those tests which deal with some particular situations and all requirements involved, just performing a simple query. The easy discovery of those test descriptions which are affected by a change of requirements could be useful to identify the test set to be re-executed or to be invalidated. However in this case some issues arise. In particular:

- It is difficult to link all requirements to a test, due to the fact that tests can cover some features, but could affect in part others requirements.
- An explicit link or mention to the requirement that is covered, intentionally or not, could lack.

The integration of semantic reasoning and text analysis techniques is very useful to support our activities. Semantic techniques can be used to resolve ambiguity and improve precision in semantic annotations or any kinds of metadata are available. On the other hand a text search engine is able to extract all keywords from requirements, which are described in natural language and which have not been explicitly included into metadata. Furthermore semantic reasoning can be used to infer from annotations and keywords, by ontologies, some implicit relationships which are not easy to be recognized. The Semantic Reasoning also demonstrates inconsistencies in language as commonly used and unconventional use of logic to overcome this issue. Usually the activity is performed by engines that derive additional assertions which are entailed from some base of knowledge together with rules and constraints associated with the reasoner. This method supports the use of formal o semi formal languages to allow additional facts to be inferred from original specification or data.

There are different types of reasoner that could be used, generally it can be classified based on the type of data accepted as input and the actions performed. The two major types of reasoners are the description logic reasoners and the rule based reasoners. The rule-based expert system [26] is used for providing answers to some expert needed problems. Usually they are strongly configurable by the user. The engines are characterized by quality and quantity. More entailments are present, then more semantics can be handled and thus more the system is complete. But the number of inference rules affects performance. Quality refers to the implementation aspects of these rules. DL reasoner works as consistency checker. It takes a document as input and returns one word being Consistent, Inconsistent, or Unknown. In others words it checks if there are contradictory facts. It support concept satisfiability, class hierarchy creation and realization in order to find the classes to which an individual belongs with the major probability. The semantic reasoner that we have to chose has to support both a standard knowledge base and a rules based approach. In order to apply our approach to different domains and to use it different methodologies, the semantic reasoner have to support a standard knowledge base representation that should guarantee a good level of abstraction, in our case ontology, and should be widely adopted. Furthermore the engine has to support the rules based approach, because also if the approach aims at being more general as possible, there is the need to customize the inference model to domain specific features.

4

A supporting tool for testing and validation of railway systems

Here we presente technologies, architecture design and usage work-flow of a prototypal tool that supports the activities of engineers in the specified phases of the safety life-cycle. Such tool can help to reduce the effort spent in different phases of the verification and validation process without any regressions of reliability and safety of the railways systems.

4.1 Architecture Design

In order to have a portable architecture, that could be integrated in different life cycles and that can be reused in different domains, we build independent components that could be integrated in an RTP (Reference Technology Platform) and connected on a bus as standalone tools. This chose was made to support sustainability and productivity. All people, vendors and associations can contribute to common objective, they can share and reuse the technology results. The approach followed in the architecture specification aims at reducing the platform proliferation, choosing one model to follow and use standard representation for data exchange. The integration with others tools and methods helps our work to be easy incorporated in others processes and others domains with a low cost of integration. An architecture based on a RTP provides an integrated set of service to the end user that is in continuous development. For example the RTP tool chain developed during the CESAR [23] project was the first platform certified by the ARTEMIS-IA and supports the development of safety critical embedded systems in the domains of automotive, aerospace, automation and rail. The coordination activities

for the so-called Cooperation RTP are under the management of EICOSE. Another RTP platform is EICOSE CRTP by ARTEMIS in which a set of system engineering assets like Technology Briks and Building blocks are stored and managed. It is an improvement started from the Cesar RTP during the Crystal project [5]. The two key concepts of a good technology platform are:

- All stakeholders have to share and cooperate following the same targets of interoperability and adherence to the chosen standards. The cooperation has to be thought of long term type.
- The RTP and the blocks which compose it have to be assessed concerning the principles defined and should be compliant both to actual implementation and the future one of the commercial products.

In order to be accessible and improvable the relevant background information should be all described into reference documentation, including the complete strategy and policy. In Figure 4.1 is shown the complete architecture. The components are all installed on the RTP bus. The components upon which the system rely are listed below:

Document Database : the document database is used to store different types of artifacts, from the requirements documents written in natural language to the test cards. The database should have good performance since we need to use not only for off.line operations, but also on-line.

Test Database : this database will be different because of different kind of performance requirements. Due to the composition of the data stored Graph Database could be used a. The test logs are in large number and have to be stored time stamped data, Boolean events, and tags. It could be queried or used to populate graphic trends.

Knowledge Base : it is used to store complex information, that include structured and unstructured ones. Its data represent facts about the word and must allow the inference engine to work. In particular the representation chosen must guarantee a good level of description, but at the same time it must sure that the inference engine will be able to finish the work and to present a valid results. In our case the knowledge base chosen is an ontology.

Text Engine : it provides full text search in different types of formats as the rich documents. We need this engine to retrieve documents or sections of it based on the words or little strings.

Indexing Engine : the indexing engine is used together with test search engine to categorize the documents and for an easy and quickly retrieval information. The indexing engine that we need should be capable to work both on-line and off-line.

Semantic Engine : the semantic engine should be able to use rules and other forms of logic to deduce new facts or highlight inconsistencies. It should allow to explore the knowledge base and infer it.

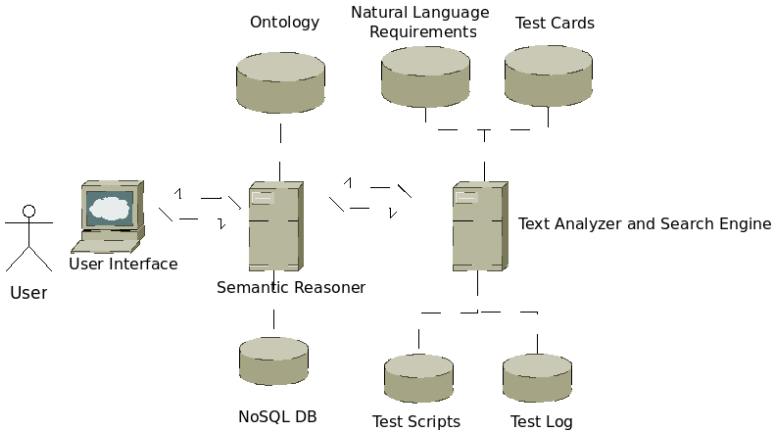

Fig. 4.1. Architecture of the proposed approach

The *Text Analyzer and Search Engine* is used for the text elaboration of documents. It has to support the retrieval of documents or information which return by the user's query. It supports dynamic clustering, database integration, and browsing of the documentation. The *Semantic Reasoner* supports the automatic test case generation. It takes the in input the requirements, which describes the functional behavior of the system, and produces in output the test scripts which fulfill defined conditions and starts from an initial state. The requirements to use as input are those already transformed and indexed by the *Text Analyzer and Search Engine*. The majors features of the *Semantic Reasoner* are the browsing of ontology and the inference of new knowledge based on the input documents and the existing knowledge base. It supports the semantic analysis of the documents and the retrieval of the information based on semantic similarities in conjunction to user rules and

constraints. It allow for the annotation of documents and for the computation of similarities between documents and artifacts in general. For example documents and test scripts could be annotated and semantically compared.

4.2 Technological framework

The technological choices are driven by both performance and interoperability criteria. As mentioned earlier the interoperability is a key attribute to consider in order to integrate the developed approach in RTP and in different domains. Among the technologies that satisfy the required characteristic the standard ones are chosen.

4.2.1 An OWL Ontology

The Web Ontology Language, or OWL, is the standard proposed by the W3C in order to define ontology for the semantic web. He defines three different semantics: OWL Lite, OWL DL and OWL Full. They are characterized by a different level of expressiveness, from the less expressive to the best one. In particular OWL DL supports the users who want an high level of expressiveness, but without losing computational completeness (There is the assurance that all conclusions are calculable) and the decidability of inferring systems (all computational operation are performed in a finite time). OWL DL contains all the construct of OWL standard, but with some restrictions a that about the type separation (a class cannot be an individual or a properties, such as a property cannot be an individual or a class). The name of OWL DL came from its agreement to the description logic, a research branch that studied a decidable subset about first order logic. The OWL DL was designed to support the section related to the description logic and it has needed computation properties for the reasoning systems.

Like all other ontology the fundamental elements of the OWL are Simple Classes, and individuals, Simple properties, properties characteristic a property restrictions. The Classes are the most basic concepts in a domain and they are the roots of various taxonomic trees. Every individual in the OWL world is a member of the class, the *owl:Thing* is the super class defined by OWL, all others class are a sub class of it. Each individuals has to belong to

a class, if there is not class defined for an individual. OWL define also the empty class *owl:Nothing*.

The first step to build a domain ontology in OWL is to define root classes, defining a named class and all each subclasses. An example of domain classes for the powered two-wheeler transport mode in the ontology could be defined the class:
<owl:Class rdf:ID="CubicCapacity"/ >
<owl:Class rdf:ID="Brand"/ >
<owl:Class rdf:ID="PoweredTwo− Wheeler"/ >

At this point each of this class can be populated with the owl individuals, in others words are declared the members of that class. For example if we define the class the class University some members of this class will be:
<University rdf:ID="Universit degli Studi di Napoli Federico II" / >
<University rdf:ID="Seconda Universit degli Studi di Napoli" / >

However the classes created and their individuals are a simple taxonomy, to give them a meaning and assert general facts we have to define the properties. There are two type of properties in OWL: *ObjectProperty* and *DatatypeProperty*. The ObjectProperty is a binary relation between two classes, instead a DatatypeProperty is a binary relation between instances of classes and RDF literals and XML Schema datatypes.

OWL DL supports also others features, but since these are out from our scope of work, to obtain a complete dissertation about OWL refer to the W3C standard [38].

We chose to develop a railway ontology for the current case study using the DL formalism. It represents the knowledge about this specific domain through a set of concepts described in a structured dictionary consisting of classes, relations, data properties and individuals. Classes (Entities) are entities that recur into the railway application. They can be both physical and abstract. e.g. the signaling equipment installed on the railway track (balise group), or the distance before a signal where it needs to react. Relations describe logical connection between two entities of the railway domain. The relations describe something that an entity has or something that an entity is. For example the entity balise is linked by the *contain* property to the *Telegram* concept. It means that in a signaling entity of type balise a Telegram is stored. Axioms

describe elementary relations, such as the sub-class between concepts or the equivalence.

In the next chapter we will describe how the owl formalism is applied in order to build a domain ontology to support the description of requirements written in natural language about the domain of our case of study. In such a way we intend to overcame the problems about the natural language and give support to the automation of the others activities.

4.2.2 Formalization of requirements

The formalization of the specifications is based on a semi formal language that describe the requirements through the use of boilerplates. They have been developed as an alternative to free text.

The availability of such semi formal description for requirements allows us for a more effective text analysis and indexation of documents. In order to allow for automatic generation of text scripts a semi formal language has been defined. At the same the semi-formal language and a domain ontology allow for semantic classification and discovery of the requirements. Therefore the system requirements are transformed into a semi-formal notation, with a well defined structure, grammar and semantic. The requirements are translated into a XML format through a finished tag vocabulary that could be used according to pre-configured structures. The vocabulary is divided into two main set of tags: the resources and the empty structures. Resources are the objects of the railway domain and requirements describe their functions and properties. Resources of this language are the main class entities of our ontology. Functions and properties are semantic object properties and data properties. Empty structures are connected using logic operators (or, and , then) in order to construct the requirements in the form: pre-condition then main, or main then post-condition. An example of empty structure build to formalize the requirements is reported below:

```
If <entity><relation><entity>
  then
     <entity>shall<relation><entity>
```

The tags elements, between the minor and major symbols, are the empty tags that can be filled. Considering a simple example about the automotive domain, where the engine shall be warmed in order to avoid carburetion problems, the template presented above can be filled with the auxiliary of an ontology in the following way:

```
if <Engine> <Uses> <Diesel Fuel>,
   <User> shall <Warm>
      the <Engine>
```

Using such a language automatic test data generator is able to take in input the set of requirements. It recognizes the structure of the requirement in terms of pre-conditions to be set in that scenario and conditions to be verified to be compliant with the requirement.

The implementation of the semi formal language presented above is based on the XML formalism. XML [40] (Extensible Markup Language) is a W3C standard to exchange a wide variety of data on the Web and elsewhere.

In the XML is possible to manage versioning and definition of the data. Furthermore there is the possibility to validate XML documents using DTDs and Schema. The solution to these problems is obtained defining a set of rules to create XML documents, DTDs and Schema, in order to share objects avoiding consistency issues.

In particular the development of XML files is performed through the following steps:

- The DTDs XML is developed starting from the data model.
- The Schema XML is developed from the data model.
- The document XML, compliant to the Schema, is written down starting from the object instance.

The main aim of the XML is the exchange of models(or data) using different representations, as for example the interchange in XMI [29] of UML models between different kind of tools. However if the schema to follow is not defined before it could be difficult to achieve full compatibility .

Another XML advantage that we exploit is the use for the automatic generation of test script and documentation. We can conclude that XML introduces a good level of abstraction and supports the reuse of the produced artifacts .

4.2.3 Text Based processing

For document analysis and indexing we used Apache Solr. $Solr^{TM}$ is a fast open source enterprise search platform from the Apache Lucene^{TM} project. Its major features include powerful full-text search, hit highlighting, faceted search, near real-time indexing, dynamic clustering, database integration, rich document (e.g., Word, PDF) handling, and geospatial search. Solr is written in Java and runs as a standalone full-text search server within a servlet container such as Jetty. Solr uses the Lucene Java search library at its core for full-text indexing and search, and has REST-like HTTP/XML and JSON APIs. Searches are done via HTTP GET on the select URL with the query string in the q parameter. You can pass a number of optional request parameters to the request handler to control what information is returned.
$q = video\&fl = name, id$ returns only $name$ and id $fields$ of the documents containing the keyword "video".

Faceted search takes the documents matched by a query and generates counts for various properties or categories. Links are usually provided that allows users to "drill down" or refine their search results based on the returned categories. The following example searches for all documents (*:*) and requests counts by the category field cat:
$...\&q = *:*\&facet = true\&facet.field = cat$

4.2.4 Transformation Templates

The requirements translated into a semi formal notation and exported in a XML format can be easy manipulated and processed using ad hoc program with embedded translation code or defining style sheet. Our choice follows the latter approach since it allows to have different templates that can be used both stand alone or integrated in a complex framework if needed. The technology chosen to transform the XML files that describe the requirements is XSLT. The Extensible Stylesheet Language Transformation (XSLT) [39] is a language by the W3C, used in order to transform XML documents in others XML documents, but with a different notation, or in others types of documents like code or documents readable by human. The term transformation is used to denote that the operation performed through XSLT means that a new artifacts is produced without modifying the source document.

The XSLT transformation process includes:

- one or more XML source files;
- one or more XSL stylesheet module;
- an XSLT template engine;
- one or more output documents.

Usually the XSLT processor [39] take two documents as input, an XML source file and a XSLT stylesheet, and starting from these write the output document. The XSLT style sheet contains the code of a XSLT program that is itself a XML document. The template is a rules collection, instruction and directives that declare how the output have to be obtained by the processor.

XSLT is a declarative programming language, it does not define a sequence of action to perform. Rules are defined into the template to specify how the node/nodes of the indicated Xpath path have to be handled. The Xpath is a path described by a hierarchy of XML tags to identify a particular node. The processor follows a fixed algorithm: assuming that it has already read and processed a style sheet, it builds a source tree from the XML input document. The first node processed is the root one: it finds the operations that have to be applied on that node and performing them produces the output tree. The rules defined into the template tells to the processor how to create the nodes into the results documents, or how to process the sub nodes into the source tree as performed for the root.

The implementation of a XSLT processor can be of two categories: server side and client side. Even if a XSLT processor is available at client side by some web browser like IE and Mozzilla, their wide spread was slow. For this reason the use of XSLT 2.0 in some domains is limited. The XSLT processor can be contained also in independent products, or as components of others software as application server, Java, .NET, or can be integrated in the operations systems. For example windows XP have a MSXML3 library that include a XSLT processor. The performance of XSLT processors is improving years by years, so also their diffusion is growing. The performance of the first processors was not good, the style sheet documents were transformed in object modeled documents and the processors worked directly upon these, also the Xpath engines were not optimized. However nowadays the XSLT processors use optimization techniques derived from techniques for querying database and functional programming languages. In addition they use tree representation more efficient in order to use less memory. We tested some XSLT processors, the results are reported in table 4.2.4. Finally the MSXML6 processor results the best one.

Processor	Document Load Time ms	Stylesheet load/compile time ms	Transformation Time ms
MSXML4	27946	14.25	116218
NXSLT2 (.NET 2.0)	274	3064	35498
MSXML6	22451	7.7	140968

Table 4.1. Workstation configuration for Neo4j Tests

The XML file used for the reported stress test are about 125 Mb, that is very big compared to the usually dimension of the translated requirements upon which it has to work. The system used for the test is reported in table 4.2

The use of XSLT in conjunction with XML allows us to derive from the requirements written in a semi formal notation the condition to set into the test script. The use of such methods allows us to develop once the template for the generation and to apply it to all projects and for all requirements version. This choice help us also to decouple the templates from the overall tool and the templates between them. In such a way also if the format of the inputoutput change, the templates can be independently modified or simply interchanged. For example suppose that in a near future the requirements will be integrated with UML models, in such scenarios it is possible to change only the templates, and only the sections that describe the nodes navigation, to use the new XMI specification of the system as input to generate the test set.

4.2.5 Semantic engine and SPARQL Queries

For supporting ontology browsing and semantic reasoning we used jOWL. jOWL is a jQuery plugin for navigation and visualization of OWL-RDFS documents. Moreover it provides APIs for SPARQL_DL query. SPARQL is a language for semantic querying of a knowledge base A number of SPARQL queries are supported natively by the jOWL library. Some examples are:

- $Class(?a)$. It can be used to list all the Classes of the ontology or to check if a given term is a Class of the ontology.
- $Type(?a, ?b)$. It allows to gets the Class if the individual ?b is given, or all the individuals belonging to the Class ?a.

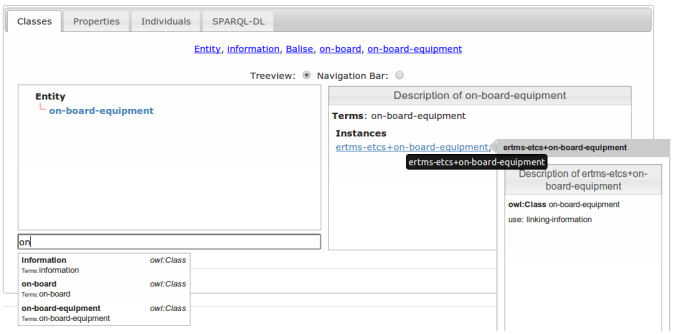

Fig. 4.2. Ontology browser

- $PropertyValue(?a, ?b, ?c)$. It returns all the properties and the correspondent values if the individual ?a is given.

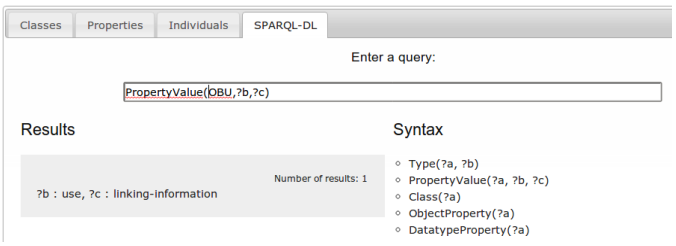

Fig. 4.3. Sparql queries

4.2.6 Big Data storage

For the implementation of the knowledge base a NOSQL solution has been chosen. The term NoSQL (meaning 'not only SQL') is used to describe a large class of databases which do not have properties of traditional relational databases and which are generally not queried with SQL (structured query language). NoSQL data stores are designed to scale well horizontally and run on commodity hardware. Also, the 'one size fit's it all' [36] notion does not work for all scenarios and it is a better to build systems based on the nature of the application and its work/data load [15].

NoSQL data stores come up with following key features [8]:

- the ability to horizontally scale 'simple operation' throughput over many servers;

- the ability to replicate and to distribute (partition) data over many servers;
- a simple call level interface or protocol (in contrast to a SQL binding);
- a weaker concurrency model than the ACID transactions of most relational (SQL) database systems;
- efficient use of distributed indexes and RAM for data storage;
- the ability to dynamically add new attributes to data records.

Most of the NoSQL databases has as the main objective, the achievement of scalability and higher performance. To provide this they do not guarantee all ACID properties, but use the a relaxed set of properties named BASE :

- Basically available: Allowance for parts of a system to fail.
- Soft state: An object may have multiple simultaneous values.
- Eventually consistent: Consistency achieved over time.

Because the data does not have to be 100 percent consistent all the time, applications can scale out to a much greater extent. By relaxing the consistency requirement, for example, NoSQL databases can have multiple copies of the same data spread across many servers or partitions in many locations. The data is instead eventually consistent when the servers are able to communicate with one another and catch up on any updates one may have missed [12]. Proponents of often cite Eric Brewer's CAP theorem[16], formalized in [6], which basically states that is impossible for a distributed computing system to simultaneously provide all three of the following guarantees: Consistency, Availability and Partition Tolerance (from these properties the CAP acronym has been derived). Where:

- Consistency: all nodes see the same data at the same time.
- Availability: a guarantee that every request receives a response about whether it was successful or failed.
- Partition Tolerance: the system continues to operate despite arbitrary message loss or failure of part of the system that create a network partition.

Only two of the CAP properties can be ensured at the same time. Therefore, only CA systems (consistent and highly available, but not partition-tolerant), CP systems (consistent and partition tolerant, but not highly available), and AP systems (highly available and partition tolerant, but not consistent) are possible and for many people CA and CP are equivalent because loosing in Partitioning Tolerance means a lost of Availability when a partition takes

place. So, the trade-offs are complex and the NoSQL databases, acting as distributed systems, must choose between either support: AP or CP.

A graph databases [34] has been chosen to improve performance figures as the semantic query will be performed natively. The choice of Neo4j is supported also by some tests executed, in particular are reported the results obtained for two types of test:

- In the first case is simulated the creation o 1000 nodes by 100 users, the results is reported in fiugure 4.4. In this case only few requests have been served in more than 800ms.
- In the second case the number of users is double, so we have the creation of 1000 nodes by 200 user. As shown by the figure 4.5 the larger number of response are in the range [800, 1200] ms, however all the requests are served.

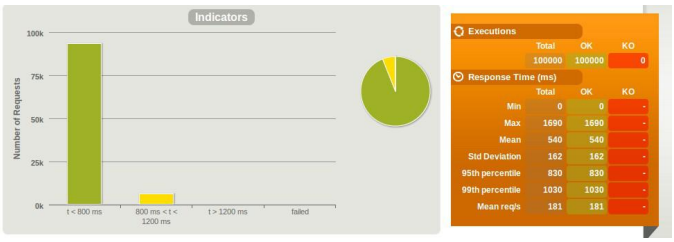

Fig. 4.4. Creation of 1000 nodes by 100 users

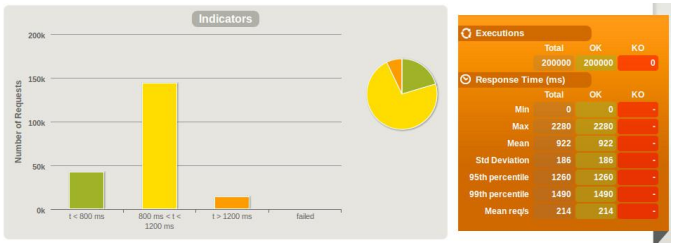

Fig. 4.5. Creation of 1000 nodes by 200 users

The configuration of the workstation used to perform the test is reported in table 4.2

Table 4.2. Workstation configuration for Neo4j Tests

Element	Description	Note
CPU	Core i5	3337U 1.8 Gz
RAM	6 GB	
SO	Windows7	64bit

Neo4j[28], is an open source, robust (fully ACID) transactional property graph database. Due to its graph data model, Neo4j is highly agile and blazing fast. For connected data operations, Neo4j runs a thousand times faster than relational databases. Nodes store data and edges represent relationships. The data model is called property graph to indicate that edges could have properties. Neo4j provides a REST interface or a Java API. The core engine of Neo4j supports the property graph model. This model can easily be adapted to support the LinkedData RDF model, consisting of Triples. Besides it is possible to add spatial indexes to already located data, and perform spatial operations on the data like searching for data within specified regions or within a specified distance of a point of interest. In addition classes are provided to expose the data to geotools and thereby to geotools enabled applications like geoserver and uDig. Adoption of will provide custom API's and Query Languages and many support the W3C's RDF standard, including a SPARQL engine. This model can easily be integrated with the JoWL base front-end.

4.3 Usage Work-flow

The tool supports the users activities for requirements analysis, test definition and test analysis.

In the following paragraphs we describe how the user interact with the proposed system to complete his work.

4.3.1 Test Definition Work-flow

A it is shown in Figure 4.6, the first step of the user is to upload the documents or the test cards on server and performs the text analysis. After that the artifacts are indexed and stored into the database. This task is the base for the activity of the semantic analysis and documents retrieval. Starting from this

documents, written in natural language, the user performs the translation into a semi-formal notation. For each requirements is identified the appropriate structure and it is filled with the support by the domain knowledge base. Once the requirements are all translated the user can perform the analysis with the help of the semantic reasoner that highlights some incoherence, lack of clarity and redundancy. For each requirements the user can brows the concept related to it, in order to understand better its meaning and the relation with the domain.

Furthermore the user is able, starting from a concept expressed into the requirements, to retrieve documents or sections that contains those elements. The user submits semantic queries to the on line reasoner that use as base of knowledge the ontology. In such a way the user is supported to retrieve all the information that needs to analyze the requirements.

Moreover this work flow is used also to verify the test coverage and the requirements traceability matrix. Starting from a retirements the user can identify all the concepts of the domains that are present and can perform a research using them as key and should limit the search only to the document categorized according to the type (g.e. all the functional test case). If the concepts used as key are too general, the research can be refined joining more concepts to obtain a single key more restrictive. In such a way the user can manage the information present in the different documents and discover the hidden relations among them.

The work flow described above is performed online with the user interaction. The system support also a off line inference based on the user action to enlarge the knowledge base and to update the connection between the documents present into the database and the last inserted by the user. In the first case the domain knowledge is enriched working on the user rules defined to make some operation upon it.

In the second case the last documents uploaded are indexing and tagged with the elements of the domain present into the knowledge base. After are researched the link between the new documents processed and the artifacts already stored. In this way are constructed the link between documents and sections in order to improve the online research performed by the user.

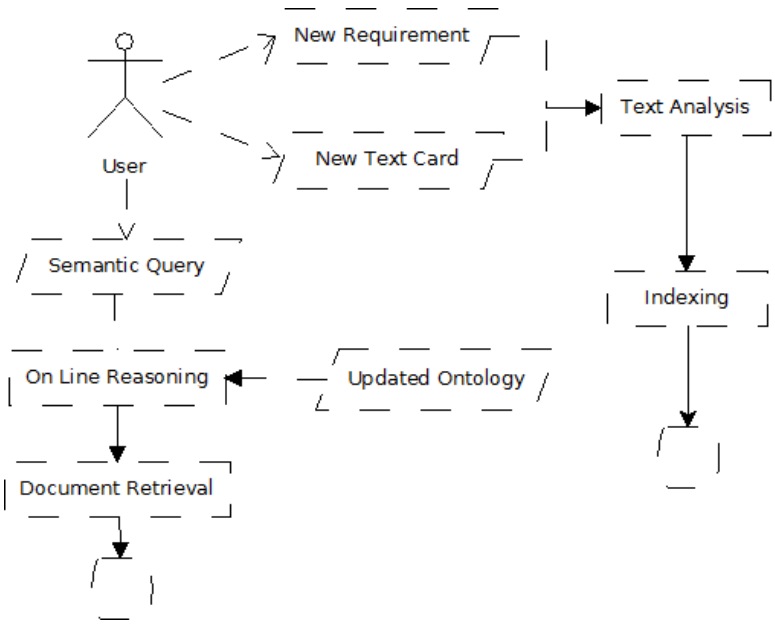

Fig. 4.6. Work flow of the verification ad validation proposed

4.3.2 Test-Log Analysis Workflow

If the requirements are stable and the test scripts are executed, the resulting test logs have to be analyzed and the issues regarding the failure have to be discovered. This activities is one of the most heavy and the user need to be supported since the large number of variables to manage and control.

In the right branch of the figure 4.7, the user can submit the test script to be analyzed, indexed and after is performed the semantic analysis and all the operations are annotated semantically with RDF triples to support the future reasoning and stored in a NoSQL database. Always in the right branch the user submits the test script for the execution in the simulation environment. The output of this activity is the test log set that can be uploaded on the server for the text analysis, the indexing and the semnatic analysis as performed for the scripts. Also in this case the test logs are annotated with RDF triples and stored in the NoSQL database. At this point the user can execute the semantic queries, performing the online reasoning on the RDF Triples that were used to annotate the test scripts and the test logs. In this way are retrieved all the documents related to the executed tests and their test scripts. This documents can be both requirements specification and similar test logs. The retrieval of

requirements specification is used to collect all the information about the test and understand the behavior of the system in that situation. The retrieval of others test logs with the same concepts could be used to better understand the errors that have caused the failure of the test and study the test analysis already performed.

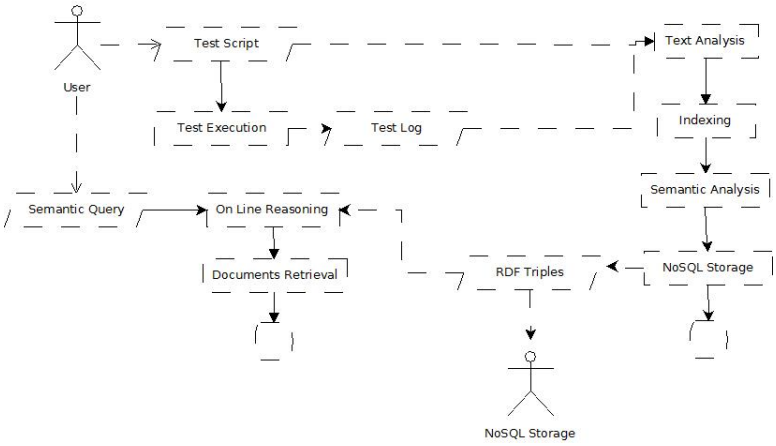

Fig. 4.7. Work flow of the verification ad validation proposed

5

Case Study and Proof of Concept

In this chapter we describe the fundamental of the railway domain that is the industrial context in which our work is developed. In particular we will describe some standard related to the railway system to explain the issues and the constraints that we need to deal with. A description of the basic behavior of the ERTMS standard is reported to give an idea about what it means to shoot a train on the rails at over 300 km/h, in a absolute safety condition.

In the second part the experimental activities will be presented and how the technologies presented in the previous chapter have been applied for the verification and validation activities about the RBC (Radio Block Centre) system. In particular we will focus our attention on some UNISIG requirements related to the signalling function of the balises.

5.1 European Projects

The European Union supports the European industry in the key sectors of the embedded systems, recognizing the centrality of automotive, aerospace, railway, and telecommunications amongst others. For this scope it funds many projects that have the aim to ensure safety and competitiveness in the global market. Even if the approach followed in such projects, as Cesar and Crystal, was general and common to all the domains involved, our attention was related to the railway domain.

This work starts from the result and the experience of the Cesar Project, in particular the early phases of process development were studied and the problem of lack of requirements formalization. During the project problems

such as misinterpretation, interference between levels, consistency and the earlier identification of errors in order to reduce work were addressed.

5.1.1 Cesar Project

The Cesar project (Cost-efficient methods and processes for safety relevant embedded systems), had the scope to provide methodologies and processes in order to cut down the cost for developing safety critical embedded systems. All the target environment of this project, automotive, aerospace and railway, had the same need about reliable embedded systems in order to satisfy the mobility request of the market, growing always more in competitiveness and globalization. In order to guarantee the competitiveness of European companies Cesar aim is to increase efficiency of development, security process and certification of embedded systems. This scope was investigated introducing in each domain at least one significant innovation in design, integration and validation process. The innovation were collected providing a software environment called RTP (Reference Technology Platform) to support development, validation and verification of requirements and architectures. This RTP embed both methodologies and tools studied and experimented during the project. The CESAR project started in March 2009 and has successfully been finished by end of June 2012 after 40 months of project duration.

5.1.2 Crystal Project

The project CRYSTAL (CRitical sYSTem engineering AcceLeration), by ARTEMIS, target is to create an European Standard for an Interoperability Specification (IOS) and a Reference Technology Platform (RTP) to address the problems that affect safety critical systems. The ARTEMIS is an Industrial Association composed by companies, universities and research institutes. It is a research group created to coordinate a set of projects in order to support the advancement of the embedded systems development. ARTEMIS-IA was founded in 2007 and now it is composed of more than 200 members. From ARTEMIS-IA the ARTEMIS Joint Undertaking was created in order to support the development of ARTEMIS European Technology Platform together with European Commission and several Member States. The ARTEMIS-JU can be viewed as Executive Manager or a Project Manager that have the scope to manage the funds allocated to a particular project and to follow his life.

The Crystal project started in 2013 and goes over the predecessor projects CEASAR, SAFE, iFEST, MBAT and in three years it aims at being very industrial oriented, indeed one of his scopes is the development of a tool chain truly usable by companies. The tool chain is developed paying attention to the support of data interoperability and information exchange through the using of standard web technologies. The data interchange is addressed not only between the different phases of the life cycle of the same domain, but also between different life cycle of the various safety industrial domains. The industrial considered domain are automotive, aerospace, rail and health. All these domains have a similar process for development of safety critical embedded systems, but in each company and for each domain a large number of COTS tools are used. These are tailored made and do not support interoperability. Many times such tools are not from the market, but are self made and so are specific for the particular activity and sometimes also for the specific project. In this way the process avoid the collaboration between different stakeholders of the domain and between the different domains. Since the interoperability between tools in the safety critical embedded systems life cycle is getting more and more critical the solution to create ad hoc bridges to support the interoperability in a single life cycle is no more applicable because the needed bridges is growing exponentially in number and in complexity, and so the related expense. The main technical challenge of the ARTEMIS Joint Undertaking project CRYSTAL is the proposal of an Interoperability Specification (IOS) and the creation of a Reference Technology Platform (RTP) to support the interchange of data between loosely coupled tools. This result is pursued using standard and open Web technologies to reduce the complexity of the entire integration process. The effort of CRYSTAL project spent on activities related to system analysis, safety analysis, system exploration, aims at improving the market of the European companies and organization on a global level. The challenging target of the CRYSTAL Project could be achieved only by the collaboration of the industrial world with tool vendors and academia, with the support of the European Countries that allows to grow the budget till 82 million.

5.2 The ERMTS Case Study

The ERTMS/ETCS stands for European Railway Traffic Management System/ European Train Control Systems and it is a signaling standard for the interoperability of railway traffic among the European rail network. The scope of the ERTMS/ETCS project is to simplify, improve and develop an international railway transport service that overcomes the national transport signalling and facilitates the standardization of products and services. The ERTMS/ETCS project is creating an open and competitive market for the suppliers and vendors and to establish a shared procedure for the assessment of conformity to interoperability requirements. The proposed architectures and technologies will increase the competitiveness of European vendors of the railway domain also on the global market, reducing equipment costs and bring the industrial to the best technology delivered by the state of art. The rail operation will be optimized in a Erope-wide scale increasing the efficiency related to environment and safety issues.

The ERTMS standard is divided into four different levels characterized by implemented equipment and functionality :

- Level 0: a ETCS train is used on a non ETCS route. The on-board unit supervises the ride only in terms of maximum speed , and the driver has the responsibility to respect the national rules and the trackside line signals.

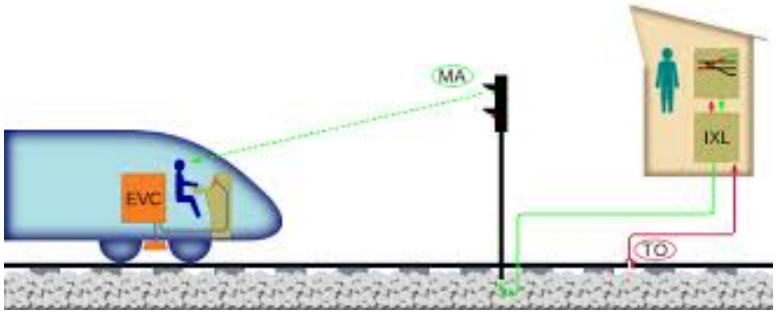

Fig. 5.1. level 0

- Level 1: is a signalling system where spot data are transmitted to the on-board unit from the trackside. This signalling system can be superimposed on and interfaced with the existing one. The equipment of the ERTMS/ETCS level one is composed by Eurobalises that send information to the on board unit at its transitions.The EuroLoop is the functional extension of the Eurobalises over a particular distance which allows data to be transmitted semi-continuously to the vehicle via a leaky cables emitting electrical radiation.

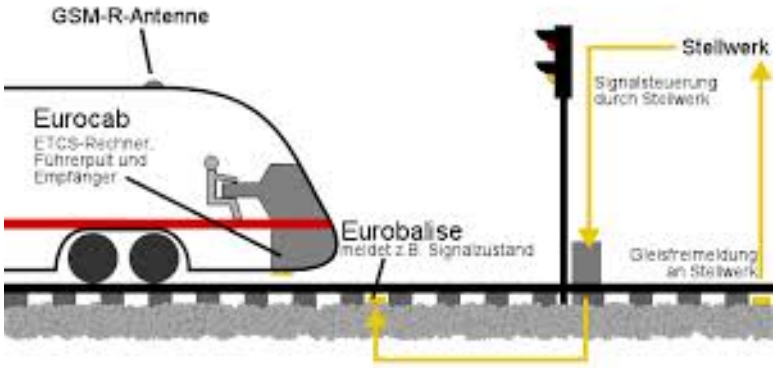

Fig. 5.2. Level 1

- Level 2: this is a train protection system based on a radio message communication between the on board unit and the central apparatus with interoperable cab signalling and fixed block sections. All information regarding the ride are exchanged via the GSM-R radio message between the on board unit and the Radio Block Centre (RBC).
- Level 3: as in level 2 the train is able to know its position due to the use of GPS signal in addition to the presence of Eurobalises and via sensors as axle transducers. In ETCS level 3 the reliability is improved so much that the train can determine its own integrity on board.

As the level 3 is not yet implemented, we focus our attention on ERTMS/ETCS level 2 that is the ETCS level implemented on the current HS/HC lines in Italy e.g. Napoli-Roma and reported in figure 5.5.

The main component that compose the architecture are the:

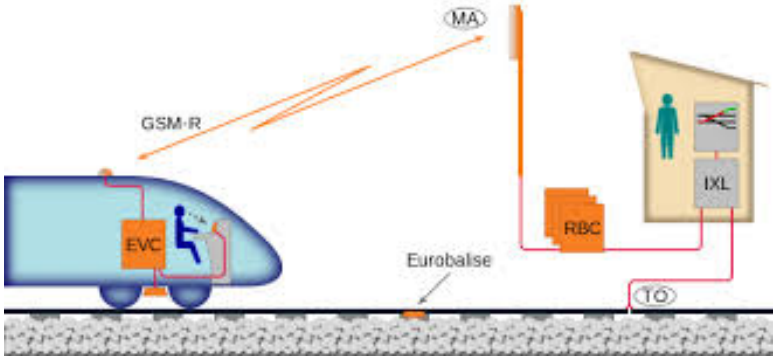

Fig. 5.3. Level 2

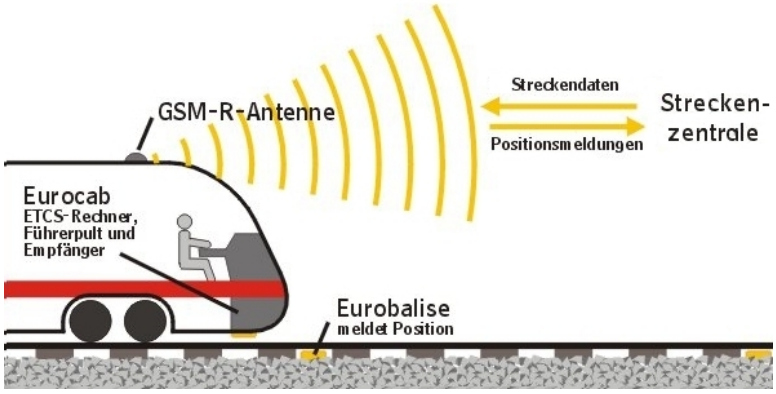

Fig. 5.4. Level 3

- Wayside Subsystem: the fixed equipment installed on the ground field and control centre equipment.
- On Board Subsystem: the control computer installed on the train locomotive.

The wayside subsystem and the on board one communicate in bidirectional way trough GSM-R messages. The European Vital Computer (EVC) sends to Radio Block Centre (RBC) the position information and the requests to perform some operation (e.g. the start of mission). The RBC replys to the train with information about the lines (e.g. the maximum permitted speed and some restriction) and the movement authorities (MA) till a fixed point. The RBC chose which MA must sent to the on board unit considering also the information received by the Interlocking (IXL). The IXL feels the track occu-

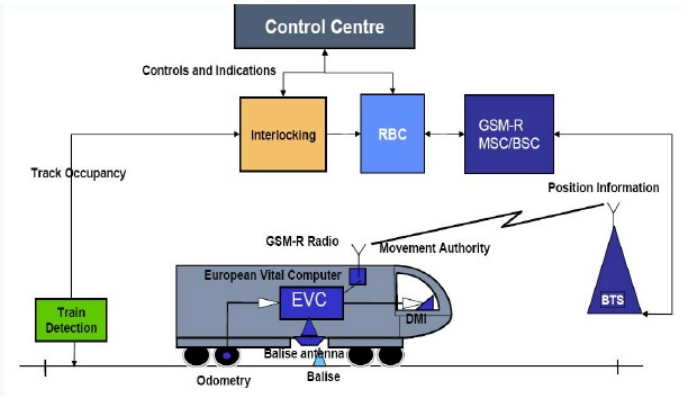

Fig. 5.5. Details of Level 2 architecture

pancy by the sensors installed in line and through actuators could command the switch point, and create itineraries. The train can receive information about the line also by the Eurobalises that are transponder installed in field, usually present in group of two object. The Eurobalise are electromagnetically excited when the train travel over them and release a Telegram capt by the balise antenna installed on board. The Telegram contains information such static speed profile, gradient information and position values to adjust the odometer error. In this operative way no light signals are used, these are replaced by the cab DMI (Driver Machine Interface) that shows all information received by the EVC and useful to the operator to supervise the train ride.

The (Supervisory Control and Data Acquisition) sub system, integrated with ERMTS/ETCS equipment, handles the regulation and the supervisor of trains movement along the line in a centralized way. Typically such subsystem provide functionality for the remote control or automatic control, in order to support, by a single human operator, the management of different stations from one place, to track trains and control railway signals, to set automatically routes and align the train ride to timetables; integrating and super visioning the entire network. Other functionalities are the monitoring and the diagnosis of the state of the infrastructure through different kinds of sensors and subsystems distributed along all the line (Passenger information, statistical data collection, etc.). The SCADA subsystem could receive also

alarms from the network (warning lights or acoustic alarms), from sensors or other subsystems.

In order to have an idea about the large number of parameters to be monitored and analyzed by such system we can think that for a network of middle high complexity there are 10.000 objects that can change state in real time, just concerning the binary sections and the trains moving along the line. In addiction to this, consider that the controlled points could be over 5.000. There are more than 3.000 signaling objects and about 1.500 text string used to communicate.

5.3 The UNISIG Specification

UNISIG is an industrial consortium funded in 1998 at the specific request of the EU Commission to develop the ERTMS/ETCS technical specifications and actively contributes to the activities of the European Railway Agency in order to assure the interoperability that is the main driver for ERTMS in the context of the European Railway Network. The founder members of the UNISIG consortium are the six majors vendors companies that operate in Europe: Alstom, Ansaldo, Bombardier, Invensys, Siemens, Alcatel(Thales) and they are called the Full Members.

The ERTMS specifications are made public and freely available by the ERA, and are accessible to any company. The detailed technical work of UNISIG is carried out in Work Packages (WPs) responsible for specific technical specifications (eg Eurobalise, Euroloop and Euroradio) or in Mirror Groups corresponding to ERA Working Groups where UNISIG is represented by appointed nominees.

The specification from the UNISIG consortium are divided in document called SUBSET. Each SUBSET deals with a specific subject and is divided into chapters. The specification describes in high level way the ERTMS/ETCS, giving also multiple solutions on how to implement a specific function and leaving a great range of freedom in the developing of the system. The UNISIG requirements could be mandatory if they are to be respected to be compliant to the standard, or optional if not.

One of the most representative and interesting UNISIG SUBSET is the SUBSET 26 which contains the ETCS System Requirements Specication (SRS) with the following chapters: introduction, basic system description,

principles, modes and transitions, procedures, ETCS language. The document describes the so-called European ETCS kernel with its interfaces to the signalling system on trackside and towards the train equipment on.board. Only few interfaces are defined in details with so-called Form Fit Functional Interface Specications. Some others are covered with Functional Interface Specications, while all the remaining interfaces between the subsystems and within the different subsystems are not harmonised at all. The kernel comprises the ETCS on-board unit (called Eurocab) and the trackside ETCS equipment.

5.4 A critical example

Considering an RBC project the number of requirements for the System Specification are about 230, which become 260 after the translation. The total number of tests that we plan in order to cover the requirements traceability matrix are 336, considering both the nominal and degraded situation. Note that these tests refer only to system tests, others type of tests are implemented in order to verify and validate the system. We developed a six months long work the test campaigns, obtaining the results shown in Table 5.1:

Release	**Executed**	**Failed**	**Test Failed from Req**	**Bugs**	**Man-hours**
Baseline 1	336	93	74	19	724
Baseline 2	124	60	54	6	378
Baseline 3	70	50	23	27	262
Baseline 4	50	30	25	5	114
Baseline 5	15	6	0	6	42

Table 5.1. Tests executed for an RBC Project

In order to analyze the presented data we calculated the Spearman correlation, the results in Table 5.2 show that the correlation between the numbers of test failed because an error in the requirements and the total number of failed tests is very high, more than the test failed due to a bug.

$$\rho = \frac{\sum i(x_i - X)(y_i - Y)}{\sqrt{\sum i(x_i - X)^2 \sum i(y_i - Y)^2}}$$

Spearmans correlation coefficient is a statistical measure of the strength of a monotonic relationship between paired data. In a sample it is denoted by ρ and is by design constrained as:

$$-1 \leq \rho \geq 1$$

Release	ρ	**p-value**
Corr(Executed/Failed)	1	0.0167
Corr(Executeed/ErrorFromReq)	0.9	0.0833
Corr (Failed/Bugs)	0.4617	0.4333
Corr (ErrorFromReq/Bugs)	0.0513	1

Table 5.2. Spearman Correlation

An error in the interpretation of the requirements can be discovered after a detection of a functional or safety anomaly during the integration phase or the deployment one. The consequence is that the the vendor must perform a bug fix releasing a new software when the system is already in field, with a high cost.

Another issue is the traceability of the test case on the requirements. This activity is performed in the initial phases, but after that it is not always maintained with the right attention. When the specification are long, complicated and especially subject to changes, trace the difference on the test specification and change this last according to the variation is not so immediate and simple. This issues bring to some problems:

- If the tester discover an anomaly of the system behavior during the verification activities it is difficult to retrieve all the possible requirements concerning that aspect of the system and to understand fully what happens. This lack of information could bring to an error in the analysis of the system because some aspects of the software are not considered, or in a worst case to miss the discovery of an error.
- If a new requirement is added to implement a new function and this is not added on the traceability matrix, or simply not added correctly to all test cases, a test set , which should be changed, is not modified, loosing its contribution to system verification and validation.

Problems in the traceability add other difficulties, in addition to those already considered, to the analysis of the test log results.

To address these problems we explore same methods and techniques for requirements analysis and information retrieval. In order to overcome the limit of the techniques actually used in the process, we proposed a semantic approach to retrieve information. In particular the semantic approach proposed is based on an domain ontology as representation of the knowledge and the use of SPARQL queries and semantic engines to infer knowledge. Transferring knowledge to the machine, through the ontology, we want to teach the machine and let it to do the work for us. The computation power of the machine, together with the advantage of the semantic techniques can bring more efficiency considering the great number of variables and information of the systems studied. In the following paragraphs our solution will be presented applying the approach from the formalization of the UNISIG requirements to the analysis of an RBC test log.

5.4.1 Railway Ontology

A railway ontology has been developed for the current case study. It represents the knowledge about this specific domain through a set of concepts described in a structured dictionary consisting of classes, relations, data properties and individuals. Classes (Entities) are entities that recur into the railway application. They can be both physical and abstract. e.g. the signaling equipment installed on the railway track (balise group), or the distance to a signal before which it needs to react. Relations describe logical connection between two entities of the railway domain. The relations describe something that an entities has or something that an entity is. For example the entity *balise* is linked with property *contain* to *Telegram*. It means that in a signaling entity of type balise a Telegram is stored. Axioms describe elementary relations, such as the sub-class between concepts or the equivalence. Going deeply the class entities are divided into twelve main classes:

- *Action*: represents an activity performed by an entity. It produces a change of the status and/or a response of the system.
- *Capability*: describes a function provided by a component of the system.
- *Event*: represents a set of external conditions or inputs. On their occurrence an action is triggered.

- *Operational condition*: define all the mode under which the railway system can perform his mission.
- *Parameter*: includes all the parameters characterizing the state of the system, on the basis of which an operational condition can occur.
- User: represents the stakeholders of the railway domain. We needs to consider not only the operators and the external subsystems that have a functional role in the system mission, but also the customer of the railway infrastructure, who could deviate the nominal behavior of the system.

In the Figure 5.6 part of the ontology developed is presented. The extract shown focuses on the class *Balise Group*, that is a real signaling entity installed along the line. This entity is composed by one or more transponders *Balise*, which have stored inside a set of information about the line properties. Information is communicated to the On Board Unit (*SSB*) of the train through a *Telegram*. All these information are represented into the picture as a graph where classes are drawn as rectangles and relations and axioms between them by edges. Different kinds of edges mean different types of relations. The arch between *Balise Group* and *Balise* represents a *Contain* relation. The same occurs between *Balise* and *Telegram*. The arch between *SSB* and *Balise* means that the train subsystem can *Capt* the balise transponder. The ontology has been written according to the W3C OWL-RDFS (Ontology Web Language - Resource Description Framework) standard.

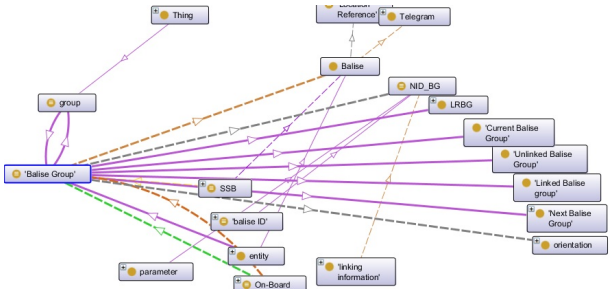

Fig. 5.6. Railway Domain Ontology

5.4.2 System Requirements

The SUBSET-026 describes, amongst the others, the requirements for linking information about balise. Each balise group has in his telegram the informa-

tion about which are the next balises, composing a balise group, that the OBU (On Board Unit) have to catch and in which direction is expected. If during his mission the train does not retrieve the right information, as expected, it needs a reaction to this event with defined actions. The following requirements are extracted by the UNISIG normative about the linking information. They describe the behavior of the OBU regarding the use of the linking information about balises. The linking information are notice present in the Telegram of the balise about the next balise group that the Train has to come across during the run.

Such example of requirements from UNISIG standard are shown in the first column of Table 5.3. They are expressed in natural language.

REQ ID	**Natural Language Requirement**	**Formal Language**
SUBSET-026-3 3.4.4.4.1	When no linking information is used on-board, all balise groups shall be taken into account.	if<SSB><use><linking information> and if<linked balise group list><contain><Current balise group>, <System> shall <consider> the <Current Balise Group>
SUBSET-026-3 3.4.4.4.2	When linking information is used on-board, only balise groups marked as linked and included in the linking information and balise groups marked as unlinked shall be taken into account.	if<SSB><use><linking information> and if<current balise group><have><unlinked marker>, <system> shall <consider> the <Current Balise Group>

Table 5.3. Examples of requirements for test definition

The requirement 3.4.4.4.1 asserts that if the OBU is in operative mode that does not use linking information, all telegrams provided by any balise noticed in line should be processed. All information provided must be taken into account in order to control the train ride. The requirement 3.4.4.4.2 instead describes operational mode using the linking information. In this case not all information taken from the balises must be processed by OBU. Only

the telegram of the balises present into the linking information list have to be taken into account. Balise marked as unlinked must be considered as source of information too. Instead the message read from balise of type linked, but not included into the list must be discarded.

To translate the requirements in the appropriate form must be create a template and initialize it with the correct values of each indeterminate part (i.e. *parameter*, *state*, *system*, *verb*, *entity*). The final statement is produced by this template.

For the requirements of the example the configurable structure to apply is:

```
 3.4.4.4.1
%if <state>, <system> shall <action>
   If <entity><relation><entity>
     then
        <entity>shall<relation><entity>
```

In the first requirement the tag $< entity >$ is initialized by the *SSB* concept and *linking information* becomes the argument of the relation *not use*, forming the precondition of the requirement. The main condition is configured with concepts *System*, *Current Balise Group* and the relation *Consider*. The resulting formal requirement is

```
if <SSB> <not use> <Linking Information>,
   <system> shall <consider>
      the <Current Balise Group>
```

In the same way the requirement 3.4.4.4.2 is obtained in semi-formal notation. First of all the requirement is divided into two different sentence 3.4.4.4.2_*a* and 3.4.4.4.2_*b*, both of them have the same structure. The structure applied is composed of two precondition in AND and of the post condition. The splitted requirements are:

a) When linking information is used on-board, only balise groups marked as linked and included in the linking information.
   AND
b) When linking information is used on-board, balise groups marked as unlinked shall be taken into account.

Applying twice the structure, following template is instantiated with concrete concepts.

If <entity><relation><entity>
and If <entity><relation><entity>,
<entity>shall <relation><entity>.

The result is available in the second column of Table 5.3.

5.4.3 System Testing

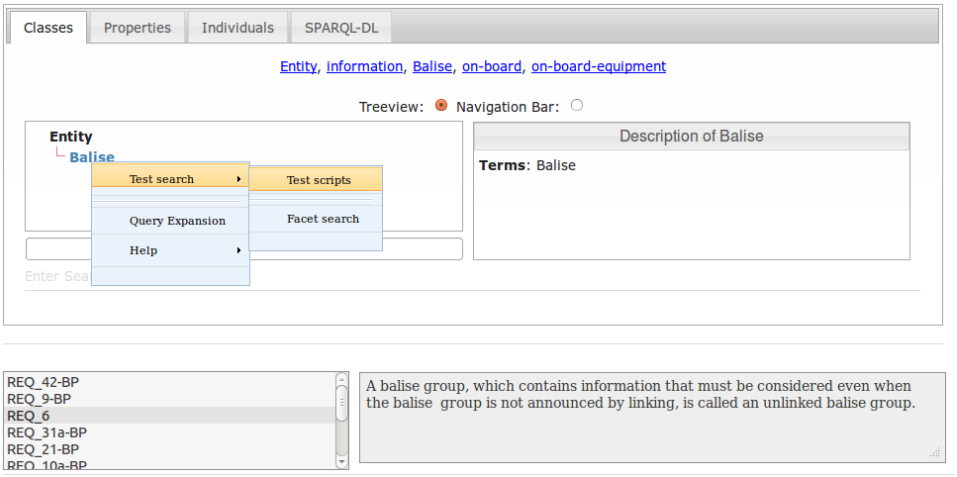

Fig. 5.7. Document retrieval

When a new project is developed it needs to define a new test plan from the requirements of that specific railway system. The test coverage must be compliant with all the requirements. The traceability matrix shows for each requirement the correspondent test(s). If some requirements are not covered, then it needs to develop new tests or to motivate why they cannot be tested and to specify those activities which can be used to demonstrate the compliance of the project. Let us suppose that the user is going to produce a test script from a specific test specification in order to fill the traceability matrix. The test specification is a description of the state of signaling entities present in line (e.g. red signals), of the action performed by the subsystem (e. g. the train) and of the expected behavior (e. g. the ground system sends the MA

to the train). The following example explains a simple procedure describing the behavior of the train in response to some events occurring during the trip. The train must discard the telegram from the second *balise_group* and has to consider the last one due to the requirements about the linking information.

Let the test description being:
"The train is located in the first SBR and is using the linking information. The train capt the first balise_group present into the first SBR and receives the MA till the signal S3 at the end of the first SBR. The train moves through the second SBR and capt the new balise_group, not present into the linking information list. The OBU discards the Telegram present into the balise and moves ahead. The train capt a third balise_group, marked as unlinked, reads the telegram and updates the Position[.....]".

Our tool allows to the user the upload of the test description. All the elements from the ontology, which occur in the text, are recognized and underlined.

Several option are available to the user. He could start from navigating the ontology browser. He can search for some terms in the text field shown in Figure 5.7. Or he can paste the text into the web page for automatic tagging of recognized terms. In the last case it is simple to set the root of the ontology tree with the concept of interest, or to check those concepts which are relevant for a text search within the document stored. Technically user actions start SPARQL queries described in Section 4.3 which aim at searching the matching entities of the ontologies, searching relations between matching concepts and listing subtypes or super-types of matching entities. All the obtained results can be effectively used to expand text queries to the Apache Solr document store.

Let us suppose that the user is interested in *OBU*, *linking information* and asks about any relations between them. It gets some triples and selects the following one: $(OBU, use, linkinginformation)$. The user is looking for all test documents which concern such expression, but he wants also to expand the search to those documents where the similar expressions occur with others concepts equivalent to OBU. Hence the system will replace OBU with on-board equipment, SSB, ERTMS-ETCS on-board equipment. Finally the SOLR back-end is queried using the following triples (SSB, use, linking information), (OBU, use, linking information), (on-board equipment, use, linking information), (ERTMS.ETCS on-board equipment, use, linking information).

In this case the requirements shown in Table 5.3 are retrieved and the user can choose those which will be included into the traceability matrix, eventually marked for review if someone is affected by an incoherence between the test and the current requirement.

5.4.4 Test Script Generation

We want to test the correct behavior of the RBC in sending the movement authority to the train to regulates its ride.

In this case we consider, in addition to the linking information explained in the requirements analysis, the SoM (Start of Mission) function derived from UNISIG normative. The requirements about SoM describe the procedure followed by on board unit (OBU) and wayside equipment in order to start the ride of the train on a ERTMS/ETCS level 2 line. We consider in particular the following requirement:

When the OBU performs a start of mission, it must send a valid SoM Position Report to the RBC. If RBC detects the position of the OBU as valid, it shall send the MA (Movement Authority) to the OBU. If RBC detects the position of the OBU as invalid, it shall not send the MA to OBU and shall activate the emergency procedure: the RBC must send an Emergency Brake Order to OBU.

In addition from the linking information we know that the train has to catch two following balise groups to be correctly located. If one of the two balise groups telegram is marked as unlinked and the train use linking information, an additional telegram from a valid balise must be read in order to have a valid position.

We can consider the following scenario to be tested in order to verify the compliance of the system with the selected requirement.

The train is using the linking information and does not know its position, because it is starting its ride and it has not yet catch any balise. Then train capts the BG A, marked as linked, and update the linkig information list. The train move ahead and catch the second balise group, but since this one is marked as unlinked, it discards the telegram. Then the train moves in position X and performs the Start of Mission. During SoM, the ETCS On board sends a SoM Position Report to the RBC and capts the balise A'. The position reports

sent before the some are not valid, since the train is not properly located. As the Position Report is invalid, the RBC could consider the train in a wrong place and could deliver a wrong MA. The RBC does not send the MA to the OBU, but it sends an invalid position alarm and the order to brake immediately. As example for our case of study we present the following abstract test script for the SoM (start of mission) function with a train that use linking information:

```
//set the state of all entities in line
//to default
For each entities in network set:
State[i]= initial_state[i]
//Force the state of  entities
//to the wanted value
For all entities in the location defined:
State[j]= setStateTo[j]
//Stimulate the system component with signals
For each Input in I
stimulate Component[i] with Input[i]
//Monitor of the output and checks
For each Output in O
check Output of O[i] equals to condition C[i]
//check the use of linking information
check Train[i] use Linking Information
Stimulate Train [i] with input [k]
// stimulate with SoMcommand the OBU
stimulate Train[i] with Input[MakeSom]
//Monitor if Som is  performed
check OBU send SoM Position Report to RBC
check RBC send MA to OBU
[]
```

In the first part of the script the state of the system under test is declared. This is a set of values characterizing the railway network and the internal state, and came from the preconditions of the requirements. Other sections of the test script include an input sequence and an output sequence. The input sequence is a list of stimuli sent to the system, they could be command sent by an operator to HMI subsystem or a change of status forced to simulate an anomaly behavior of the system. The output sequence is a set of data or control actions which are produced by the system in response to a certain input. The

output sequence is automatically obtained from the post conditions defined into the requirements. For each state, input sequence and output sequence, some checks are declared to monitor the right execution of the test. The checks operate also as break point, if the result of the test is False, the next item of the sequence (input or output) is not scheduled. When this test is executed into the simulated environment it fails because the RBC did not send the MA to OBU, because the train uses the linking information and failed to be located because it discarded the second telegram. In particular the test log produced can be resumed by the following code.

```
//set the state of all entities in line
//to default
//For each entities in network set:
Time 353 State[i]= initial_state[i]
Time 354 State[i+1]= initial_state[i+1]
[..]
//Force the state of  entities
//to the wanted value
//For all entities in the location defined:
Time 555 State[j]= setStateTo[j]
Time 556 State[j+1]= setStateTo[j+1]
[...]
//Stimulate the system component with signals
//For each Input in I
Time 767stimulate Component[i] with Input[i]
Time 777 stimulate switch point 32
      position C_B
[...]
Time 778 set Linking information to train

//Monitor of the output and checks
//For each Output in O
Time 888 check Output of O[i]
      equals to condition C[i]
Time 889 check position  switch point 32
      equals to C_B  True
Time 890 check if Train use Linking Information True
[...]
// stimulate with SoM command the OBU
```

```
Time 999 stimulate Train with Input[MakeSom]
//Monitor if Som is  performed
Time 1000 check OBU
     send SoM Position Report to RBC TRUE
Time 1001 check RBC send MA to OBU FALSE
Time 1002 test failed
Test Stopped
(No others operations are executed)
```

The output of the test execution is a set of test logs that include the report of test script, the monitor of all variables about simulated subsystems and the output of the stimulated one. The log is the output of the script executed in which time-stamps and the value assumed by variables to check are reported. Therefore the log contains the state, the input sequence as reported into the original file, but with the time stamps. For what concerning the output sequence and the associated checks, the time stamp and the value assumed by the output are reported and, in case of error, the notification of failure. Note that if a failure is detected the test is declared not passed and following input or output items are not processed. The report of the executed script is a simple way to verify the test result and the termination mode, but an accurate analysis is essential, by an investigation about what happen in each subsystems, in order to understand the cause of failure.

5.4.5 Document analysis

$Solr^{TM}$ has been used to extract keyword for cited documents and their score. Such keywords are matched with the domain ontology to infer RDF triples which semantically describe each document. Taking into account the example described in Section 5.4.2 we infer the following metadata from UNISIG SoM (Start of Mission) and Linking Information requirements.

```
(OBU, use, LinkingInformation)
(OBU, perform, SoM)
(OBU,is, located)
(OBU, send, SoM Position Report)
(RBC, send, MA) (OBU, Recive, MA)
(RBC, send, EmergencyBrake).
```

In the same way the description of the considered scenario can be described by the following semantic annotation.

```
(OBU, use, LinkingInformation)
(OBU, send, SoM Position Report)
(RBC, send, MA)
(OBU, Receive, MA)
```

Of course such information will be linked to the the related document and between them. Finally the failed log of our example is processed. Each check instruction of the output sequence from the last control block is semantically represented. From the following two lines:

```
Time 890 check OBU1 use Linking Information TRUE
Time 1000 check OBU1 send SoM
    Position Report to RBC1 TRUE
Time 1001 check RBC1 send MA to Treno1 FALSE
```

then we infer the following triples:

```
(OBU1, use, LinkingInformation)
(OBU1, send, SoM Position Report)
(RBC1, receive, Position Report)
(RBC1, send, Ma)
(OBU1, receive, MA)
```

The log is explicitly linked to the correspondent requirements and scenario in the knowledge base, both to the instance of the ontology and to the text document in the Solr tool.

5.4.6 Log Analysis

During the log analysis, when a failed log has been detected, the user can look for similar failures in the knowledge base in order to understand the errors. The user can use the tuples from failed log shown in the previous example to search which test are failed due to checks on the same variables of the same entities.

One among available documents, resulting from the query that uses the tuples describing the log presented before, contains the following lines.

```
Time 332 check Linked balise group list
     contains  ETCS5233 FALSE
Time 333 check OBU1 send SoM
     Position Report to RBC1 TRUE
Time 334 check RBC1 send MA to Treno1 FALSE
```

represented by the following tuples:

```
(Linked Balise group list, contain, ETCS5233)
(OBU1, send, SoM Position report)
(RBC1, send, MA)
```

The found test log has an additional check if it is compared to our example. Retrieving the correspondent test description is easier to understand how to find the cause of the failure. In this particular case the additional check allows to detect that the Train sends a Position report when it is not correctly located because the second ETCS is marked as unlinked and so its telegram is discarded by the OBU.

5.4.7 Results

We have performed the verification and validation activities, following the approach proposed above, and with the support of the developed framework on an experimental project; in particular we have tested the baseline 4 of paragraph 5.4, with 10 tests, 5 of which failed due to errors from requirements and 1 from a bug. The results are reported in Table 5.4:

Release	**Executed**	**Failed**	**Test Failed from Req**	**Bugs**	**Man-hours**
Baseline 4	10	6	5	1	21.5

Table 5.4. Tests executed for an RBC Experimental Project with proposed approach

In the Figure 5.8 the diagram that compares the trend of the failed test following the two approaches is reported. The series related to our approach is in green, the old one in blue. Instead in Table 5.5, the projection of the total amount of hours saved is reported, considering that from the previous example to execute 10 test with the analysis of 6 faults we spend 21.5.

The comparison shows that the proposed approach bring an improvement of 5,7% in terms of time spent. This is a cheering result that supports the adoption of our approach in the real industrial process.

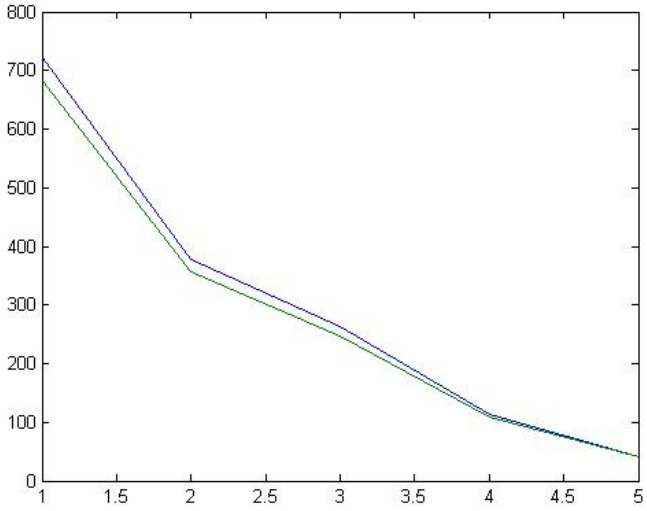

Fig. 5.8. Failed Test trend

Release	**Man-Hours Case 1**	**Man-Hours Case 2**	**Saved Time**
Baseline 1	724	683	41
Baseline 2	378	356	22
Baseline 3	262	247	15
Baseline 4	114	108	6
Baseline 5	42	40	2

Table 5.5. Improvement in hours for tests execution and analysis

Conclusions

We discussed about the critical issues that affect the verification and validation activity in the life cycle of Safety Critical Embedded systems. In particular we focused on the requirements analysis and management and on the test definition activity that are affected by the usage of natural language for the requirement specification in terms of effort, correctness and coverage. We proposed the integrated utilization of semantic techniques and text search engines for supporting the users to improve traceability of requirements and test descriptions. We introduced a methodology to support the test logs analysis using semantic techniques and full text search.

We presented a framework for management of requirements, test scripts and logs that are produced during the simulation of the system for the validation and verification activities. The framework uses a graph database for storing documents and their properties. It provides a web dashboard by which the user is able to browse the knowledge base starting from a domain ontology or interactive controls that execute semantic and keyword based queries.

It also allows for an advanced discovery and retrieval of test descriptions and requirements. Preliminary development of a web tool that demonstrates the proposed approach in a real case study has been presented.

The support system proposed for the verification and validation of reliable industrial products have lead to improve the efficiency of the RAMS activity performed to guarantee the adherence of the ERTMS/ETCS system to the defined requirements and safety target. Even if the effort spent in the early phase for the translation of the requirements from the original natural language to a structured notation with well defined grammar and semantic was

greater if compared to the time and costs of the original phase. The benefit achieved are greater than the effort spent for the additional effort introduced.

At the end of this work we are able to highlights the main issues that are still open and the improvements that can be addressed in order to close some open points and extend the range of problems considered. In particular some open issues still affect the prove of the correctness about the translation between the requirements written in natural language and the structured ones. A automatic support to the verification of this phase should be introduced, since it is demanded to user and it is not free from errors.

A key activity to be implemented is the integration of the approach proposed to the Model driven development according to the V-Model defined in the CENELEC EN 50126. In particular the overall life cycle can be take a great advantage introducing the model driven approach in the both development and validation side of the process, as remarked in [7].

Another key methodology that if introduced can bring to good results in terms of quality of the product developed are the formal methods and in particular the model checking. These techniques allow us to verify LTL properties about a system, but since the model checking investigated at the moment use explicit-state model checker, and may not scale to large data domains due to state-space explosion, only the most critical sub components of the systems can be validated by this technique.

In the recent years these methods have not gained a great success. In particular the companies prefer to use consolidated process in which they trust instead to introduce new technologies and methods. In the always more competitive global market is essentials for the industrial world invest into research and do not postpone the application of innovative methodologies since a delay of few time could bring to lost wide market segments and could be irreparable. A better synergy between the academic research world and the industrial is essential to achieve good results and to finance innovative projects. The successful of the industrial system itself depends by the results that are achieved in such application, so is important not only prove their benefit with theoretical consideration, but also give evidence in the system development.

References

[1] A. Amato, B. Di Martino, and S. Venticinque. A semantic framework for delivery of context-aware ubiquitous services in pervasive environments. pages 412–419, 2012.

[2] Alba Amato, Giuseppina Cretella, Beniamino Di Martino, and VENTICINQUE S. *2013 27th International Conference on Advanced Information Networking and Applications Workshops*, chapter Semantic and Agent Technologies for Cloud Vendor Agnostic Resource Brokering, pages 1253–1258. IEEE CPS, USA, 2013. ISBN 978-0-7695-4952-1. doi: DOI 10.1109/WAINA.2013.163.

[3] Alba Amato, Beniamino Di Martino, Marco Scialdone, and VENTICINQUE S. *Intelligent Distributed Computing VII*, volume 511, pages 261–270. Springer International Publishing Switzerland, 2013.

[4] F. Amato, A. Mazzeo, A. Penta, and A. Picariello. Building rdf ontologies from semi-structured legal documents. In *Complex, Intelligent and Software Intensive Systems, 2008. CISIS 2008. International Conference on*, pages 997–1002, March 2008. doi: 10.1109/CISIS.2008.146.

[5] ARTEMIS. Crystal - critical system engineering acceleration. URL `http://www.crystal-artemis.eu/`.

[6] Eric A. Brewer. Towards robust distributed systems (abstract). In *Proceedings of the nineteenth annual ACM symposium on Principles of distributed computing*, PODC '00, pages 7–, New York, NY, USA, 2000. ACM. ISBN 1-58113-183-6. doi: 10.1145/343477.343502. URL `http://doi.acm.org/10.1145/343477.343502`.

[7] Gabriella Carrozza, Mauro Faella, Francesco Fucci, Roberto Pietrantuono, and Stefano Russo. Engineering air traffic control systems with a model-driven approach. *Software, IEEE*, 30(3):42–48, May 2013. ISSN 0740-7459. doi: 10.1109/MS.2013.20.

[8] R Cattell. Scalable sql and nosql data stores. Technical report, 2012.

[9] CENELEC. Railway applications the specification and demonstration of dependability, reliability, availability, maintainability and safety (rams). Technical report, .

[10] CENELEC. Railway applications: Software for railway control and protection systems. Technical report, .

[11] European Commission. Eu transport in figures, 2013.

[12] DataStax. A guide to big data workload- management challenges. Technical report, 2012.

[13] Ofer Egozi, Shaul Markovitch, and Evgeniy Gabrilovich. Concept-based information retrieval using explicit semantic analysis. *ACM Trans. Inf. Syst.*, 29(2):8:1–8:34, April 2011. ISSN 1046-8188.

[14] Miriam Fernandez, Ivn Cantador, Vanesa Lopez, David Vallet, Pablo Castells, and Enrico Motta. Semantically enhanced information retrieval: An ontology-based approach. *Web Semantics: Science, Services and Agents on the World Wide Web*, 9(4):434 – 452, 2011. ISSN 1570-8268.

[15] Santhosh Kumar Gajendran. A survey on nosql databases. Technical report, 2012.

[16] Seth Gilbert and Nancy Lynch. Brewer's conjecture and the feasibility of consistent, available, partition-tolerant web services. *SIGACT News*, 33(2):51–59, June 2002. ISSN 0163-5700. doi: 10.1145/564585.564601. URL `http://doi.acm.org/10.1145/564585.564601`.

[17] FP6 Integrating and Strengthening the ERA programm. Integrail: Intelligent integration of railway systems. URL `http://www.integrail.inf`.

[18] Vitaly Klyuev and Vladimir Oleshchuk. Semantic retrieval: an approach to representing, searching and summarising text documents. *International Journal of Information Technology, Communications and Convergence*, 1:221–234, 21011.

[19] Reymonrod Vasquez Kunal Verma, Alex Kass. Using syntactic and semantic analyses to improve the quality of requirements documentation. Giancarlo Guizzardi.

[20] N. Lodemann, M. ; Luttenberger. Ontology-based railway infrastructure verification - planning benefits. SciTe Press , Spain, 2010.

[21] Marco Lormans. Reconstructing requirements coverage views from design and test using traceability recovery via lsi. In *In Proc. of the Int. Workshop on Traceability in Emerging Forms of Software Engineering*, pages 37–42. Press, 2005.

[22] Analia Lourenco, Rafael Carreira, Glez-Pena, Jose R. Mendez, and alt. Biodr: Semantic indexing networks for biomedical document retrieval. *Expert Systems with Applications*, 37(4):3444 – 3453, 2010. ISSN 0957-4174.

[23] et ali Mader Roland, Armengaud Eric. Integrated tool chains for model based development of embedded systems: The cesar approach. *ECMFA 2010 HoPES Workshop*, 2010.

[24] John J. Marciniak. *Encyclopedia of Software Engineering*. Wiley, USA, January 2002. ISBN 978-0-471-37737-5.

[25] MathWorks. Code verification and run-time error detection through abstract interpretation.

[26] Georgios Meditskos and N. Bassiliades. A rule-based object-oriented owl reasoner. *Knowledge and Data Engineering, IEEE Transactions on*, 20(3):397–410, March 2008. ISSN 1041-4347. doi: 10.1109/TKDE.2007.190699.

[27] F. Moscato, B. Di Martino, S. Venticinque, and A. Martone. Overfa: A collaborative framework for the semantic annotation of documents and websites. *International Journal of Web and Grid Services*, 5(1):30–45, 2009.

[28] Inc. Neo Technology. Neo4j, the world's leading graph database., 2012. URL `//www.neo4j.org/`. [Online; 26-July-2013].

[29] Object Management Group OMG. Xml metadata interchange, 2013. URL `http://www.omg.org/spec/XMI/2.4.1/`. [Online; 03-June-2013].

[30] A.V. Paliwal, B. Shafiq, J. Vaidya, Hui Xiong, and N. Adam. Semantics-based automated service discovery. *Services Computing, IEEE Transactions on*, 5(2):260–275, 2012. ISSN 1939-1374. doi: 10.1109/TSC.2011.19.

[31] P. Thrimurthy Rajasekhara Rao, Sastry KR Jammalamadaka. A semantic model for testing embedded systems. In *International Journal of Systems and Technologies*, pages 31–38, 2008.

[32] La Repubblica. Disastro ferroviario in cina morta anche un'italiana, 2011. URL `http://www.repubblica.it`. [Online; 24-July-2011.

[33] Juergen Rilling, René Witte, and Yonggang Zhang. Automatic traceability recovery: An ontological approach. In *International Symposium on Grand Challenges in Traceability (GCT'07)*, Lexington, Kentucky, USA, March 22–23 2007. Center of Excellence in Traceability, Center of Excellence in Traceability. ISBN 1-59593-6017/03/07.

[34] I. Robinson, J. Webber, and E. Eifrem. *Graph Databases*. O'Reilly Media, Incorporated, 2013. ISBN 9781449356262. URL `http://books.google.it/books?id=RTvAmQEACAAJ`.

[35] Graeme Smith. *The Object-Z specification language*, volume 101. Citeseer, 2000.

[36] Michael Stonebraker and Ugur Cetintemel. "one size fits all": An idea whose time has come and gone. In *Proceedings of the 21st International Conference on Data Engineering*, ICDE '05, pages 2–11, Washington, DC, USA, 2005. IEEE Computer Society. ISBN 0-7695-2285-8. doi: 10.1109/ICDE.2005.1. URL `http://dx.doi.org/10.1109/ICDE.2005.1`.

[37] et ali Veronica Castaneda, Luciana Ballejos. The use of ontologies in requirements engineering. In *Global Journal of Researches in Engineering*, page Vol. 10.

[38] W3C. Owl web ontology language guide, 2004. URL `http://www.w3.org/TR/2004/REC-owl-guide-20040210/`. [Online; 10-February-2004].

[39] W3C. Xsl transformations (xslt), 2007. URL `http://www.w3.org/TR/2007/REC-xslt20-20070123/`. [Online; 23-january-2007].

[40] W3C. Xml extensible markup language, 2013. URL `http://www.w3.org/XML/`. [Online; 29-October-2013].

[41] J. Wegener, K. Grimm, and M. Grochtmann. Systematic testing of real-time systems. In *Conference Papers of EuroSTAR 96*, Amsterdam, 1996.